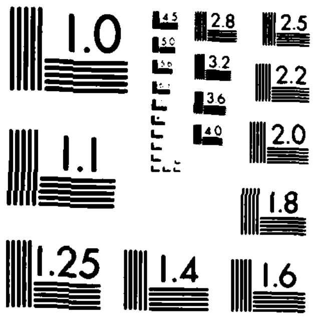
AD-A144 357 MULTI-DISCIPLINARY TECHNIQUES FOR UNDERSTANDING TIME-VARYING SPACE-BASED..(U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF ELECTRICAL ENGINEE.. 1/2

UNCLASSIFIED D CASASENT ET AL. 01 JUN 84 F/G 20/6 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFOSR-TR- 84-0597

4

AD-A144 357

DTIC FILE COPY

DTIC
SELECTE
AUG 1 7 1984
S
A

Approved for public release;
distribution unlimited.

84 08 07 113

ANNUAL REPORT

May 1983 - May 1984

MULTI-DISCIPLINARY TECHNIQUES FOR UNDERSTANDING TIME-VARYING SPACE-BASED IMAGERY

Prepared For:

Air Force Office of Scientific Research
Bolling Air Force Base
Washington, D.C. 20332

ATTN: Dr. David Fox
Director, Mathematical and Information Sciences

Prepared By:

D. Casasent, Electrical and Computer Engineering
A. Sanderson, The Robotics Institute, Electrical and Computer Engineering
T. Kanade, Computer Science

Carnegie-Mellon University
5000 Forbes Avenue
Pittsburgh, Pennsylvania 15213

Date: 1 June 1984

AIR FOR[illegible]
NOTICE [illegible]
This [illegible]
app[illegible]
Dist[illegible]
MATTHEW J. K[illegible]
Chief, Technical Information [illegible]

Table of Contents

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S) **AFOSR-TR- 84-0597**
6a NAME OF PERFORMING ORGANIZATION Carnegie-Mellon University 6b OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research
6c ADDRESS *(City, State, and ZIP Code)* Electrical & Computer Engineering Pittsburgh PA 15213	7b. ADDRESS *(City, State, and ZIP Code)* Directorate of Mathematical & Information Sciences, AFOSR, Bolling AFB DC 20332
8a. NAME OF FUNDING / SPONSORING ORGANIZATION AFOSR 8b. OFFICE SYMBOL *(If applicable)* NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F49620-83-C-0100
8c. ADDRESS *(City, State, and ZIP Code)* Bolling AFB DC 20332	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO
61102F	2304	A7	

11. TITLE *(Include Security Classification)*
MULTI-DISCIPLINARY TECHNIQUES FOR UNDERSTANDING TIME-VARYING SPACE-BASED IMAGERY

12. PERSONAL AUTHOR(S)
David Casasent, Arthur Sanderson, and Takeo Kanade

13a TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15 PAGE COUNT
Interim	FROM 1/5/83 TO 30/4/84	JUN 84	163

16. SUPPLEMENTARY NOTATION

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Artificial intelligence; image understanding; feature extraction; space-based imagery; time-varying images.

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*
A multi-disciplinary program for space-based image processing is reported. This project combines optical and digital processing techniques and pattern recognition, image understanding and artificial intelligence methodologies. Time-change image processing was recognized as the key issue to be addressed. Three time-change scenarios were defined based on the frame rate of the data change. This report details the recent research on: various statistical and deterministic image features, recognition of sub-pixel targets in time-varying imagery, and 3-D object modeling and recognition.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Dr. Robert N. Buchal	22b TELEPHONE *(Include Area Code)* (202) 767- 4939 22c OFFICE SYMBOL NM

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

ABSTRACT

A multi-disciplinary program for space-based image processing is reported. This project combines optical and digital processing techniques and pattern recognition, image understanding and artificial intelligence methodologies. Time-change image processing was recognized as the key issue to be addressed. Three time-change scenarios were defined based on the frame rate of the data change. This report details our recent research on: various statistical and deterministic image features, recognition of sub-pixel targets in time-varying imagery, and 3-D object modeling and recognition.

DTIC COPY INSPECTED

A-1

1. INTRODUCTION

1.1 OVERVIEW

This project is a multi-disciplinary effort intended to combine the following disciplines and address the following topics and applications:

1. Space-Based Imagery is our primary concern. Such data involves extensive amounts of information acquired per frame with associated image recognition, feature extraction and bandwidth compression needs.

2. Time-Change Data is our primary application. This concerns the extraction of features and the registration of subsequent image frames.

3. Three Time-Frame Scenario Definitions have been defined.

The three principle investigators are from the Departments of Electrical and Computer Engineering, Computer Science, Robotics and Biomedical Engineering. In this Chapter 1, we advance our definition of the space-based image processing problem (Section 1.2). This includes defining of specific issues associated with space-based time-sequential imagery, a definition of the necessary operations, functions, and disciplines required, and the description of three scenarios for time-change detection. We then discuss the importance of this work to Air Force technology and to related Air Force programs (Section 1.3).

In Figure 1.1, we show the general structure for our proposed hybrid optical/digital system using multiple methodologies for understanding space-based images. As shown in Figure 1.1., input images are preprocessed and then fed to parallel optical and digital channels in which multiple features are extracted. A parallel image modeling system is also shown which extracts structural descriptions of the image. These data plus image registration and target detection information obtained from an optical correlator channel are then used by an AI/IU system to modify the parallel input processing channels, to assemble and interpret a time-history track file on objects of interest in the image and to provide the necessary textural and graphic output reports. In Section 1.2, we discuss the use of this hybrid multi-disciplinary system for time-varying space-based image processing problems. The relevance of this work to Air Force technology is noted in Section 1.3. An introductory discussion to our first year of research follows in Section 1.4 with details in subsequent Chapters 2-4 and Appendices A-C.

1.2 PROBLEM DEFINITION

Advanced space-based sensor systems will provide us with *high-resolution real-time* multisensor data acquisition in the near future. This will totally pollute present processors unless we address how to intelligently and timely process and handle the projected data rates. NASA and others have already verified that the United States is capable of collecting more data than we can intelligently process [Wilson and Silverman, 1979] (less than 1% of all NASA data has even been looked at [Wilson and Silverman, 1979]).

The key issue in Space-Based Image Understanding (SBIU) is not to transmit every frame of data (with 5000 x 5000 sensor elements in three bands with ten bits of data per pixel, and a 30 frame/sec rate, this is a data collection rate of over 10^{10} bits/sec). No existing technology can accommodate this. Therefore, attention should be given to the algorithms required to achieve this. But first, let us denote several facts about SBIU problems:

1. In space-based image acquisition, we are monitoring certain areas and regions for diverse well-defined missions. We are only concerned with changes and do not need to know that nothing new has occurred in the image being looked at. When we transmit only the associated *change information*, we achieve a quite significant *bandwidth reduction*. Thus, we should process the data from space-based sensors on-board the platforms, determine image changes on-line, interpret the results and transmit only textural and graphic output reports.

2. We know rather well where the satellite is and where it is looking and we know that the scene being imaged correlates with the prior image frame or with our stored reference. The problem is thus different from the often discussed unbounded and unsupervised target recognition problem. We can and must utilize this *a'priori information* that the frame we are investigating correlates with a previous one in our processing algorithms.

3. To provide better *image registration* accuracy and to facilitate pointing of secondary sensors at given areas of interest, it is often necessary to *locate key landmarks* in the image. This is also useful in determining *geometrical corrections* needed.

4. It is also useful and necessary to register two successive image frames for *inter-frame integration* to decrease the variance of the noise and to improve the image quality. This

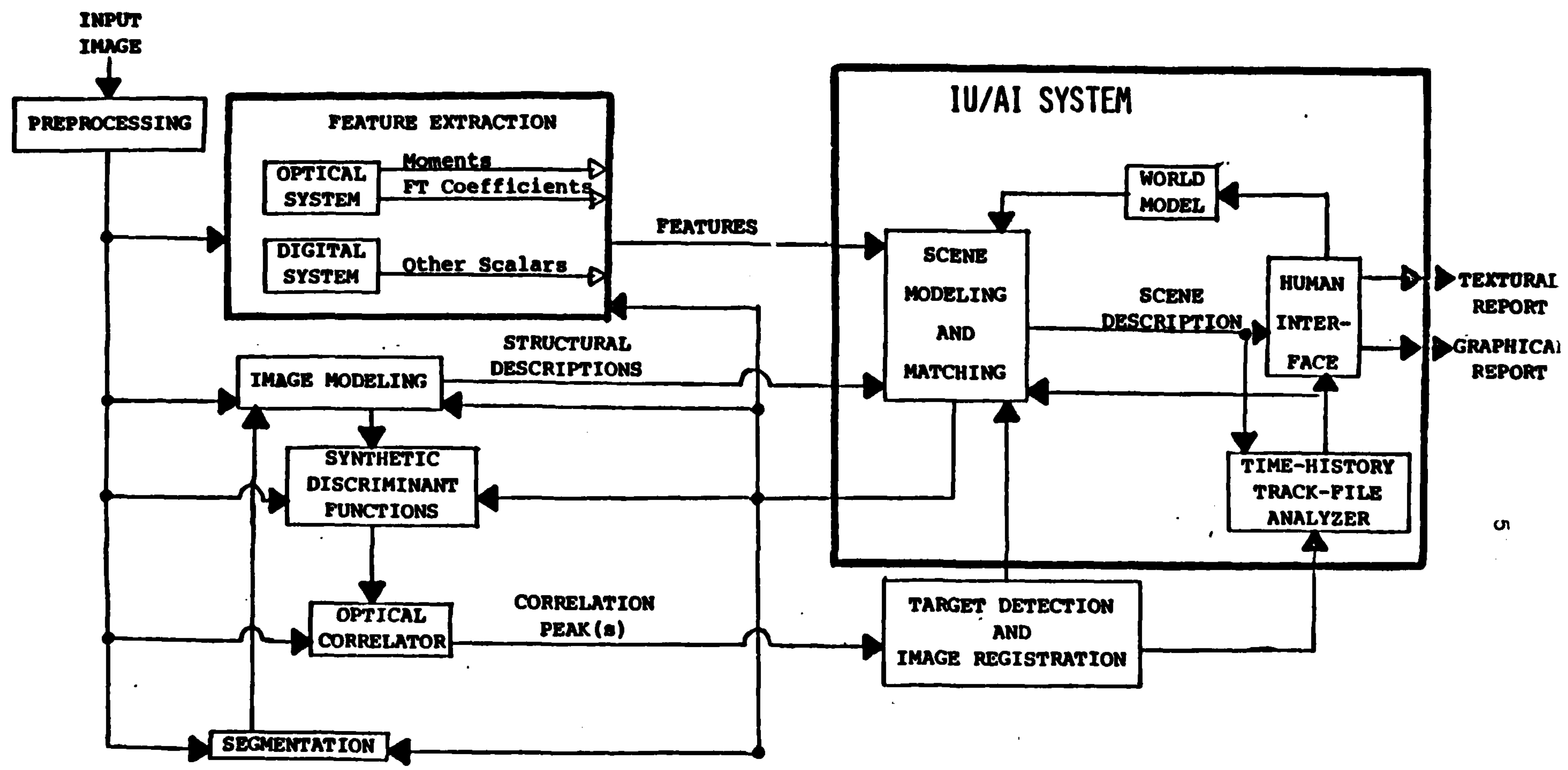


Figure 1-1: Hybrid optical/digital multi-disciplinary (pattern recognition/image understanding and artificial intelligence) processor for space-based imagery

is essential to accommodate platform variations with time and background drift. Often, *sub-pixel image registration* is necessary.

5. It is obviously essential to *subtract successive frames* since this provides the necessary change detection or time-varying target data.

6. However, in most cases, the image registration in (4) is sub-pixel and thus before performing (5), we must *interpolate* the images.

7. Once *time-history track files* of candidate objects of interest in the field-of-view of the sensors have been obtained, a multitude of discrimination analysis techniques, AI, IU, pattern recognition and human perception algorithms are necessary to classify, understand and interpret the time-change activity noted.

8. In advanced sensor systems, 3-D information on the scenes will be available from stereo satellites or other techniques. In such cases, we can fully capitalize on the available image information only by the use of advanced *3-D scene modeling and interpretation*. The key point is the extraction of scene information (3-D) from time-histories of 2-D images.

9. To detect and describe detailed changes in the 3-D structure of scenes, it is useful to first *generate 3-D scene descriptions from the 2-D images*, and then to compare the descriptions for changes. Conventional 2-D change detection approaches are not as useful for high resolution images of complex scenes since they do not take into account factors such as different viewpoints and different lighting conditions for the different images of the scene. In order to detect changes over successive images of a given scene obtained over time, it is useful to maintain a 3-D model of the scene and automatically update the model as changes occur. This requires the ability to match the model with each new view of the scene. *Matching in 3-D* is more desirable than matching in 2-D since the 3-D information is represented in a manner that is independent of viewpoint and lighting conditions.

10. Let us now briefly discuss 3-D scene information. The *3-D scene model* is a useful central component for many aspects of the change detection task. Not only is it useful for determining whether changes have occurred, but it also permits *model-based interpretation* of new images and serves as a central representation for accumulating 3-D scene information from various low-level experts. Our new research addresses these aspects of time-history 3-D scene information.

Items 1-6 address the high throughput signal processing aspects of SBIU, whereas items 7-10 address the advanced image understanding aspects of this problem. These SBIU objectives are summarized in Table 1.1, the required techniques are noted in Table 1.2 and the disciplines required to achieve our goals are enumerated in Table 1.3. In Tables 1.1 and 1.2, we also note the importance of efficient database organization and manipulation since storage or transmission of a very large database will be required for SBIU.

To properly address understanding of time-varying space-based images, we feel that three different SBIU time-varying image processing scenarios (Table 1.4) must be separately addressed. We propose to study each of these during the course of our research. We distinguish the three cases by the change rate and the domain of analysis. In the first case (rapid time-variations), we can consider a missile launch. In this application, the objective is to track the time-history of the missile and to transmit the information that a missile has been launched (from subsequent sensors, the missile's trajectory etc. can be obtained from our system techniques and algorithms). The second case (medium time-variations) can concern monitoring of key sites such as airports, railroads and harbors and know areas of anticipated concentrations of troops or armor. In this case, troop or armor movement and air, land and sea activity can be obtained from time-varying image data. This second scenario is typical of a case in which extensive AI and IU techniques are appropriate (i.e., the use of information on the locations of hangers, runways, railroad tracks, terminals, switching yards, harbor channels, docks, piers, etc.). This also requires the locations and registration of these items in sequential image frames. The third case (slow time-variations) addresses urban development and agricultural or land use activity (as in Landsat and ERTS case-studies).

Table 1-1: Objectives of Space-Based Image Processing

Detection of image changes
Use of *a priori* knowledge
Location of key landmarks
Time-history track file acquisition
Interpretation of time-history data
3-D *scene* interpretation
Efficient storage and retrieval of information from database

The three scenarios noted in Table 1.4 constitute our definition of the SBIU problem. All cases require the techniques and multi-disciplines noted in Tables 1.2 and 1.3. The first case (rapid time-variations) requires primarily sub-pixel image registration, frame integration, frame interpolation, and

Table 1-2: Space-Based Image Processing Techniques Required

Image enhancement and preprocessing

- Image registration (sub-pixel) for frame integration Image subtraction for time-history extraction Image interpolation for image subtraction

Image segmentation
Feature extraction
Image modeling
3-D scene modeling and interpretation
Hierarchical database design

Table 1-3: Disciplines Required to Achieve Real-Time Space-Based Image Processing

Pattern recognition
Image understanding
Human perception
Artificial Intelligence
Optical Processing
Digital Processing

Table 1-4: Time-Change Scenarios

TIME CHANGE	EXAMPLES	DOMAIN OF ANALYSIS
Rapid	Missile Launch	Image Pixels
Medium	Railroad, Airport, Harbor, Troops, Armor	Scene Structure
Slow	Agricultural, Land-use, Urban Development	Statistical Image Modeling

image differencing. The second case requires techniques involving image interpretation, 3-D scene modeling, 3-D matching and comparison, plus knowledge-based geometric reasoning. The third case needs more statistical techniques and statistical image models, more so than do the others. All cases require object and scene modeling, image preprocessing and enhancement plus segmentation, feature extraction and classification. Figure 1.1 depicts these aspects and the interactive multi-disciplinary feedback required to solve these SBIU problems.

1.3 BENEFIT TO AIR FORCE TECHNOLOGY

With our three scenario problem definition (Table 1.4), we now consider the myriad of Air Force programs and technology that can benefit from our proposed research. First, we note that our research is directed toward the development of new algorithms and their realization in a hybrid optical/digital architecture. However, devices and architectures being developed in related Air Force programs in VHSIC and VLSI, systolic array processors, Josephson junction devices, etc. can also be used for implementation of these algorithms. Our work will thus provide problem definition and direction regarding algorithms for such parallel processor architectures and technology programs. Large data storage requirements and studies of what constitutes a valid database are also integral parts of this program. Similar Air Force efforts toward data storage and database acquisition are thus of direct concern to this program. The Air Force programs in: intelligent sensors, intelligent task automation, automated manufacturing, image understanding, human perception and visual psychophysics will directly benefit from the inter-disciplinary nature of our research. The large Air Force effort in optical data processing will directly benefit since real-time spatial light modulators and holographic optical elements will be needed for implementation of our algorithms in real-time. The Air Force programs in missile guidance require a new set of algorithms and attention to the database requirements and performance measures used and thus they will likewise benefit extensively from this program. Darpa/AF programs such as HALO and HICAMP will clearly benefit from our chosen time-varying SBIU tasks.

The monitoring of changes and developments at cultural sites, such as urban areas and military bases, is a very useful application of space-based sensors. The techniques we develop will aid in detecting and describing both large-scale and detailed changes. Furthermore, the techniques dealing with 3-D matching and comparison, and knowledge-based geometric reasoning will enhance Air Force programs in sensing and robotics.

1.4 RECENT YEAR ONE RESEARCH

Our algorithm research (Chapter 2) has concentrated on three major approaches. All of these are suitable for incorporation of parallel optical processing technology methods. Earlier C-MU work under AFOSR support has developed optical feature extraction systems using moments [Casasent et al, 1982], chord distributions [Casasent and Chang, 1983], a Fourier coefficient WRD (wedge ring detector) sampled feature space [Casasent and Sharma, 1984]. These optically-generated feature spaces offer attractive parallel feature extraction systems which our algorithms can exploit. The use of synthetic discriminant functions [Casasent, 1984] offers the most powerful method for present use. Its incorporation is an issue for future research, although our present orthogonal basis function work (Chapter 2) is quite related.

The complexity of image processing problems is closely related to the type of representation which is derived from the image. While gray-level recognition of image fields using template correlations may be carried out on raw image data, more complex interpretation of scenes and relation of scenes to object models requires the identification of structural elements of the scene and the ability to reason about these elements in the context of the rest of the image. One objective of the current research is to define techniques which exploit the efficient preprocessing capabilities of optical and parallel processors to derive structural elements of the image, and to interactively interface such a structural representation to model-based interpretation systems. The strength of such an approach would be the ability to adaptively define the preprocessing strategy as the scene interpretation proceeds, interactively searching the tree of potential image interpretations. Clearly, such an approach requires not only exploration of new algorithms, but also the development of computer architectures which can support such tools. In the current effort we have undertaken development of a host architecture which will support an array processor for efficient simulation of such an integrated hybrid optical/digital system (Appendix A).

The robotics RAPIDBUS equipment facility assembled largely under this AFOSR support is detailed in Appendix A. The optical/digital processing facility in Electrical and Computer Engineering is described in Appendix B. This Electrical and Computer Engineering facility was supported by multiple funds, but its use is extensively devoted to DoD applications. The Computer Science equipment facility purchased under this AFOSR support is detailed in Appendix C.

Two types of elements are useful to extract from images in order to interpret objects and surface contours. *Structural elements* are typically local subpatterns of the image related to edges, corners, regions, or other features of the object shape. Shading, shadows, and object occlusion also affect the interpretation of the scene. *Textural elements* are local subpatterns which contain locally random intensity patterns and are modeled as locally random correlations among image intensities. In the approaches described here, representations are constructed from convolution and correlation operations which are particularly suited to parallel processing. Chapter 2 details our present efforts in this research area. Optical realization of these parallel algorithms is a subject of future research.

Time-varying images suggest several different modes of representation reflecting changes either in raw image elements or in model-based interpretations. While time-varying properties of the image increase the complexity of the task in the sense of data volume and degrees of freedom of the model, there are cases where motion in the scene may provide additional information particularly in regard to three-dimensional properties of moving objects. Such motion stereo is of primary interest for short range imaging, though we will investigate potential applications of these ideas to aerial imagery. Time-varying image representations fall into several major categories including: point trajectories, object trajectories, texture trajectories, optical flow, and change detection. Point and object trajectories are interpretations of the motion of structural elements in two or three dimensions. Optical flow methods use continuity and smoothness constraints on the time-varying motions to derive mathematical relations between observed changes in various parts of the image. Optical flow methods are used primarily for the interpretation of stationary scenes with a moving camera. Texture trajectories reflect the changes which occur in image properties associated with texture. For example, long term changes in foliage, crops, earth surface properties, or water surface conditions result in time-varying texture properties.

Chapter 2 describes our structural and statistical image modeling work on probabilistic graph matching, multiple resolution structural basis functions and textural surface models. These methods and others will be used to model and interpret time-varying properties of scenes. We later plan to consider the parallel processing of optics to compute element structure and texture data, plus other features. Our initial results on a high rate time change scenario requiring the extraction of a sub-pixel target are provided in Chapter 3. Our work on 3-D object modeling, time-varying 3-D images and model updating are discussed in Chapter 4.

Change detection techniques such as those derived by Segen and Sanderson [Segen and Sanderson, 1980] provide a basis for decomposition of time-varying signals into locally stationary regions, and for the detection of significant departures from current trends using sequential techniques. In the case of time-varying imagery, such techniques offer the possibility of detecting the initiation or the suspension of movement, and in longer term time-varying data allow one to model periods of constant image properties and detect deviations from the current model. The optimal choice of such models of local stationarity has also been investigated by Segen and Sanderson [Segen and Sanderson, 1979] and will be used here to identify appropriate models for slowly varying imagery. These methods of change detection will not be a focus of research in the initial phase of the project, but may provide useful tools as we proceed to more complex data.

The focus of research on algorithms during the first year has been in the development of structural and textural representation techniques which will lead to effective time-varying descriptions in the form of object or texture trajectories. From this point of view, probabilistic graph models provide a very general mathematical tool for organizing and relating structural elements to object models. The probabilistic formulation of these models incorporates mechanisms for assessing the reliability of image elements as well as the context of image elements relative to each other. Structural basis functions are intended to expand local image patterns in a multiple resolution framework such that local image properties may be recognized and efficient description of trajectories may be formulated at multiple levels. Tracking of objects at low resolution with periodic verification of the object description at higher levels is an efficient approach. Textural properties also may change with time and the relation of image texture features to surface contour properties is an important aspect of the interpretation. Chapter 2 summarizes our approach to these three issues achieved during year one.

The three time-change scenarios devised include rapid time-change (subsecond), medium time-change (involving minutes or hours) and slow time-change (involving days or months). The high rate time-change application concerns location and identification of sub-pixel moving aircrafts and missiles against a 3-D earth and cloud background.. The medium rate time-change scenario concerns monitoring airports, ship ports and troop activity. The long rate case concerns textural changes and the addition of new buildings and similar structures in militarily significant regions. Chapter 3 describes our initial successful results in extracting sub-pixel moving targets from clutter. Chapter 4 describes our initial work on producing 3-D building models and updating them from new imagery.

The technique used to extract time-varying sub-pixel moving targets involves the correlation of successive frames of data. Optics is quite attractive at achieving this. From the shape of the correlation peak, estimates of the sub-pixel shifts between two successive frames are made. These shifts occur due to platform motion, earth rotation, etc. Our image frame is then shifted by the estimated amount and subtracted from the other frame. The sequence of these operations provides a track file on moving objects to which IU and AI techniques can then be applied to achieve identification and tracking.

This requires modeling of the scene. Our initial model uses correlated and uncorrelated noise. This models clouds in the scene, sensor noise and much earth structure when viewed from a high altitude. Our generation methods for such data are proceeding well. Various estimators have been investigated to determine which is best, which requires the fewest iterations, the least computational load, which provides the largest noise reduction and the best sub-pixel shift estimate. Several interpolators have been investigated to achieve the necessary sub-pixel shift without destroying the sub-pixel target. Our initial results and the successful extraction of sub-pixel moving targets are presented in Chapter 3.

To detect and describe detailed changes in the three dimensional structure of scenes, we take the approach of first generating 3-D scene descriptions from the 2-D images, and then comparing the descriptions for changes. This is in contrast to the conventional approach of comparing and detecting changes in the images themselves, which suffers from the fact that changes in the images do not necessarily correspond to changes in the geometry and structure of the 3-D scene. There are two reasons for this. First, when a given scene is illuminated differently (by changing the positions and number of light sources), changes will occur in the shadows, highlights, and shading that appear in the image. These changes, of course, do not correspond to any changes in the 3-D structure of the scene. Second, when a given scene is viewed from different positions, the images taken at these positions will appear different because of occlusions and effects due to perspective projection such as foreshortening. Again, these changes do not correspond to changes in the scene itself.

Our overall approach involves analyzing successive high resolution images of a given scene which are obtained over time. Three-dimensional information is extracted from each image and accumulated over the multiple images to form a 3-D model of the scene. When a new image is analyzed, 3-D information may be due either to the scene having changed, or to errors in the model and/or 3-D information extracted from the new image. In either case, the model will be updated with this new information. The updated model is then used in processing the next image. Note that the scene model plays the role of a central representation. It incrementally accumulates information about the scene, and it represents the current understanding of the scene. It may therefore be used not only to determine whether 3-D changes have occurred, but also for tasks such as model-based image interpretation, planning paths through the scene, and generating up-to-date maps of the scene.

In this report, we describe results in two aspects of the change detection task as applied to high resolution aerial images of urban scenes: low-level image analysis and high-level model maintenance. The goal of the low-level analysis is to extract boundaries of buildings in the image. This is important for obtaining a 3-D description of the buildings. The first step in extracting the boundaries is to find linear features in the image. These feature may arise either from building boundaries or from other scene events, such as shadow boundaries or texture. In order to determine the source of each linear feature, we utilize a priori knowledge of the task domain. We use the knowledge that roofs of buildings tend to be parallel to the ground plane, while walls tend to be perpendicular to it. This constitutes a weak, generic model of buildings. Lines in the image which seem to fit this model are then labeled as parts of building structures. This technique makes heavy use of junctions in the image, which are assumed to correspond to building corners. However, the procedure for forming junctions can be very expensive if done naively. Efficiency may be obtained if the image is initially divided by a grid of squares. Each square, called a sector, has a list of all the lines passing through it. When we want to find all the lines passing through a small window centered on some point in the image (to determine whether these lines might form a junction), we consider only the lines passing through the sectors containing the window. In this report, we present a study that determines the optimum number of sectors with which to divide the image.

This report also discusses our techniques for representing, constructing, and updating the 3-D scene model. It is assumed here that we can extract 3-D wire-frame descriptions representing portions of boundaries of buildings in the scene. Furthermore, it is currently assumed that the scene does not change. In this way, we need not yet handle the full range of problems dealing with interpreting and handling contradictory information. In the future, we will approach the problem of dealing with changing scenes. In our current approach, the scene model is a surface-based description and is incrementally acquired from a sequence of images. We use the generic model of buildings described earlier to construct and update the scene model from the wire frames. Our 3-D building modeling and updating work are detailed in Chapter 4.

CHAPTER 1 REFERENCES

[Casasent et al, 1982]

D. Casasent, R.L. Cheatham and D. Fetterly, "Optical System to Compute Intensity Moments: Design", Applied Optics, Vol. 21, September 1982, pp. 3292-3298.

[Casasent and Chang, 1983]

D. Casasent and W-T. Chang, "Generalized Chord Transformation for Distortion-Invariant Optical Pattern Recognition", Applied Optics, Vol. 22, July 1983, pp. 2087-2094.

[Casasent and Sharma, 1984]

D. Casasent and V. Sharma, "Feature Extractors for Distortion-Invariant Optical Pattern Recognition", Optical Engineering, Vol. 23, November 1984.

[Segen and Sanderson, 1979]

J. Segen and A.C. Sanderson, "A Minimal Representation Criterion for Clustering", in Proc. 12th Ann. Symp. Comp. Sci. and Stat., University of Waterloo, May 1979.

[Segen and Sanderson, 1980]

J. Segen and A.C. Sanderson, "Detecting Change in a Time-Series", IEEE Trans. Inform. Theory., Vol. IT-26, March 1980, pp. 249-255.

[Wilson and Silverman, 1979]

R. Wilson and W. Silverman, Proc. SPIE, Vol. 178, 1979; W. Silverman and R. Wilson, Proc. SPIE, Vol. 201, 1979; NASA CP-150258, April 1977.

2. Statistical and Structural Image Modeling and Change Detection

2.1 Introduction

This phase of the project focuses on the development and evaluation of algorithms which yield representations of structural and textural information in an image, and relate these representations to object and surface contour properties of the scene. These representations will be used as the basis for modeling and interpretation of time-varying properties of scenes. The techniques currently being studied include:

- *Probabilistic Graph Matching* - Attributed graph structures are used as the basis for a composite structural-statistical model of information in an image. Structural elements such as edges, corners, regions, or local basis function expansion peaks are used as elements of the graph structure. Matching of object classes, of stereo pairs, and of sequential time-varying scenes have been studied. (Section 2.2)

- *Multiple Resolution Structural Basis Functions* - Local expansion of the gray level intensities of an image is carried out at multiple resolution levels. This hierarchical representation of the image is useful for the detection and recognition of objects, and facilitates the description or tracking of time-varying objects at low resolution with updating of object description at higher levels. (Section 2.3)

- *Textural Surface Models* - The relationship between image texture and surface contours is fundamental to the interpretation of images with textured regions and particularly to the interpretation of variations in textured regions over time. These studies are looking at the relationship between two-dimensional random field models of surface contour and observed intensity fields under known conditions of illumination and imaging. (Section 2.4)

The structural basis function and texture models described will be implemented and evaluated on the array processor with RAPIDbus host described in appendix A. This architecture will provide the basis for integration of parallel preprocessing algorithms with representations which are well-suited to model-based interpretations. The interactive use of parallel and optical preprocessing with hypothesis formation and adaptive search strategies is a longer term goal of this research.

2.2 Probabilistic Graph Matching

In a variety of image processing problems, the data contains stereotyped subpatterns which are well-described by symbolic representations. Such symbolic representations include graph, grammar, and automata models. While these models are very useful when subpatterns are highly invariant to image variability, symbolic search and manipulation techniques become very complex when symbol correspondence becomes uncertain. Symbolic representations may be enhanced in two respects which increase their applicability to real data. First, stochastic structures may be used to associate outcome probabilities with structural relations of the model. Second, attributed structures offer a rich class of models where subpatterns or symbols have associated features or attribute values. Such attributed structural models pose many difficult methodological issues for implementation. In this study we have addressed problems of the dichotomy between symbolic and statistical information and its effect on the choice of symbol primitives, issues of structural observability, structural matching, assumptions of component independence, and identification of structural transformations. These issues will be discussed in papers and reports now in preparation.

An attributed random graph model consists of a 4-tuple $R = (V,\alpha,E,\beta)$ where:

1. the *random vertex set* $V = \{ V , i = 1,\ldots,n\}$, where each V is a random variable called the *random vertex.*

2. the *random edge set* $E = \{ E , i = 1,\ldots,n, j = 1,\ldots,n\}$ in V X V where each E is a random variable called the random edge.

3. the *random vertex attribute set* $\alpha = \{\alpha_i, i = 1,\ldots,n\}$ where each α is a random variable with possible outcomes {a}.

4. the *random edge attribute set* $\beta = \{\beta_i , i = 1,\ldots,n, j = 1,\ldots,n\}$ where each β is a random variable with possible outcomes {b}.

5. Each outcome of $R = (V,\alpha,E,\beta)$ is an attributed graph $H = (v,a,e,b)$ with probability P(H) = Prob $\{V = v, \alpha = a, E = e, \beta = b\}$ such that

   - $P(H) = 0$ for all $H . \varepsilon . \Gamma$,

   - $\Sigma_\Gamma P(H) = 1$, where Γ is the range of R.

   P(H) is the *probability distribution* of R.

The attributed random graph model defined above provides a basis for the definition of likelihood functions over the observed outcomes from the class of graphs and the attribute set A. The likelihood of an observed outcome may be used as a basis for the matching and recognition of patterns in the image. In this application the structural elements of the image are associated with vertex and edge symbols of the model, and both structural relations and quantitative properties of the elements are retained in the model. Image components such as vertices, edges, regions, or intensity peaks may be used as structural elements. In the resulting probabilistic model, each element has some outcome probability, some observation probability, and some probability density of attribute values.

A simple example of a graph representation derived from a gray level image of a polyhedron is shown in figure 2.1. A line drawing of the original image is shown in figure 2.1a. The graph structure extracted from a single observation is shown in figure 2.1b where graph vertices have been attached to structural corner elements of the original image and graph edges have been attached to edge elements of the original image. An ensemble of observations such as that in figure 2.1b is used to derive a probabilistic graph model such as that shown in figure 2.1c. In figure 2.1c, the probability distribution of positions of the vertices are indicated by circles. The probability distribution of vertex angle attributes is indicated by $p(\theta)$.

Probabilistic graph matching may be used for matching of images and recognition of objects in images using likelihood criteria as a basis for search correspondence trees. The likelihood of any observed graph, subgraph, or structural element may be computed and used for estimation or decision making. Such problems incorporate three phases: (1) correspondence matching of graph elements, (2) rigid graph pose estimation, and (3) likelihood calculation. The probabilistic graph model uses pose independent likelihoods to hypothesize correspondence, then estimate pose. The use of attributes to guide correspondence matching, and the use of observation probabilities to structure the search results in simplified and reliable algorithms.

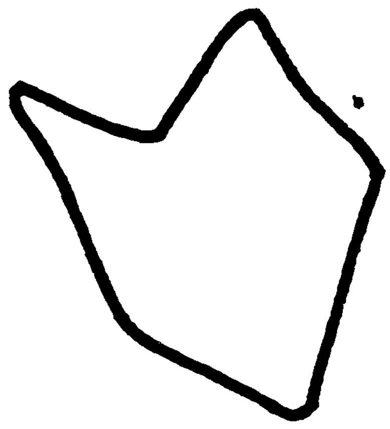
We have applied the probabilistic graph matching approach to two types of image representation. Graphs which are derived from edge, corner, and junction components of gray level images are useful for description and matching of objects. In this case, attributes include lengths, angles, and positions of elements. The resulting graph models have been used to classify objects, inspect objects, and determine orientation of objects in scenes where edge information is a reliable clue. An example of matching likelihoods between a model graph and various distorted observation graphs including partial views is shown in figure 2.2. The likelihood is a measure of the correspondence between the two structures in each case. A similar approach may be used to track movement of the model object by matching successive views and computing the pose changes between views. Such an example is shown in figure 2.3 for the same object used in the previous examples.

As a second example, we have used probabilistic graph models for matching of multiresolution tree structures derived from gray-level images. The derivation of such a multiresolution tree using the difference of low-pass transform is described in the next section. In this case, the matching algorithm is formulated as a hierarchical tree search using likelihoods to guide correspondence matching at each level of the tree. Results of this approach are described in [Crowley and Sanderson 84].

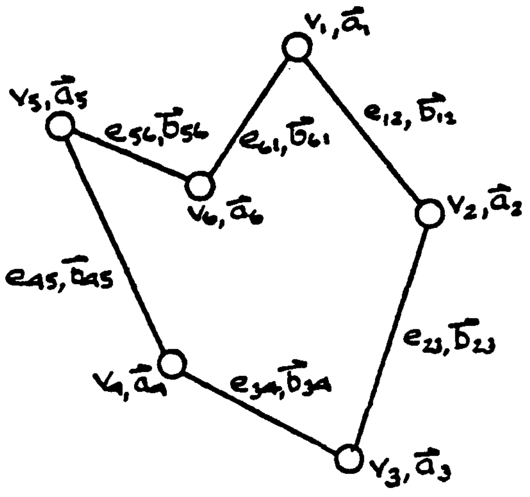
Probabilistic graph models provide a general approach to many practical matching and recognition problems where a training set of images of a priori knowledge of the probability structure is assumed.

Figure 2-1: Probabilistic graph derived from an ensemble of gray-level images.

a Raw Image Outline



b Sample Graph



c Model Graph for Ensemble

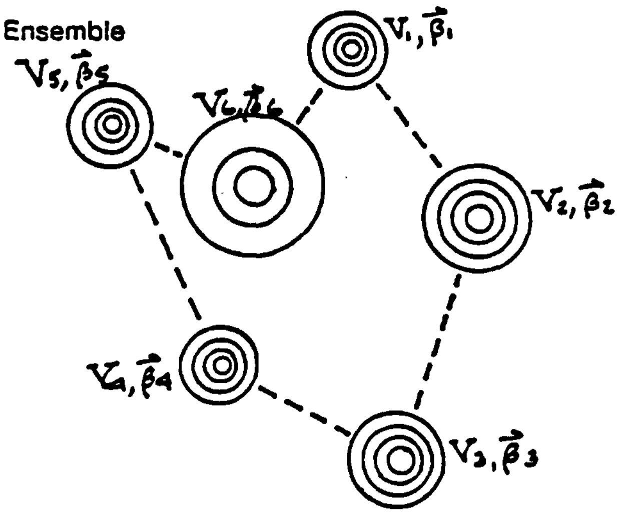
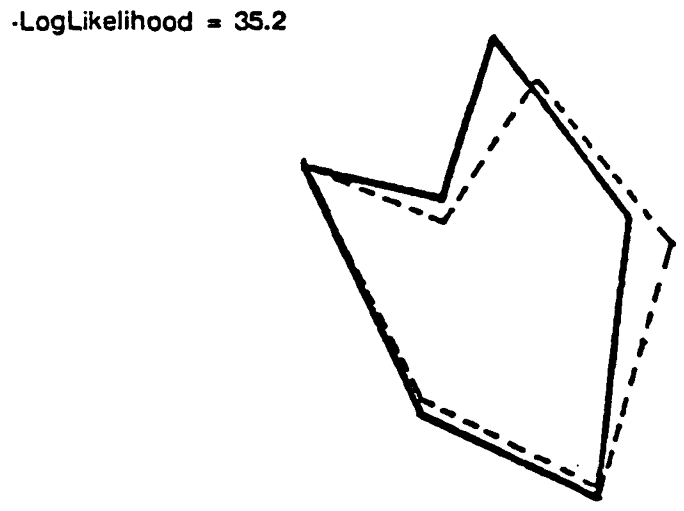


Figure 2-2: Matching graphs between a model and distorted observations with associated likelihoods.



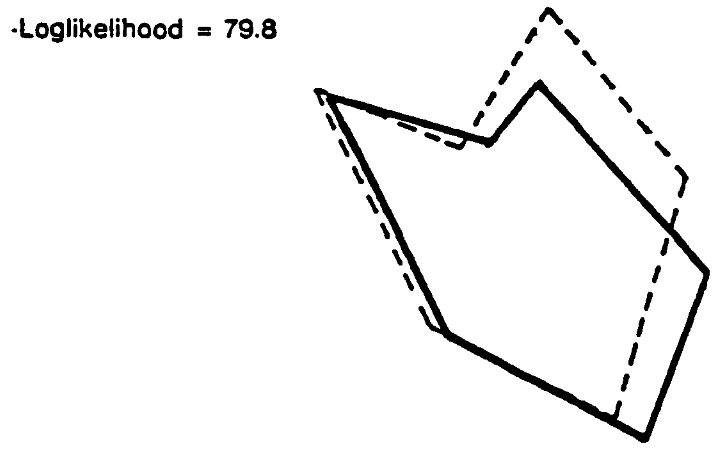
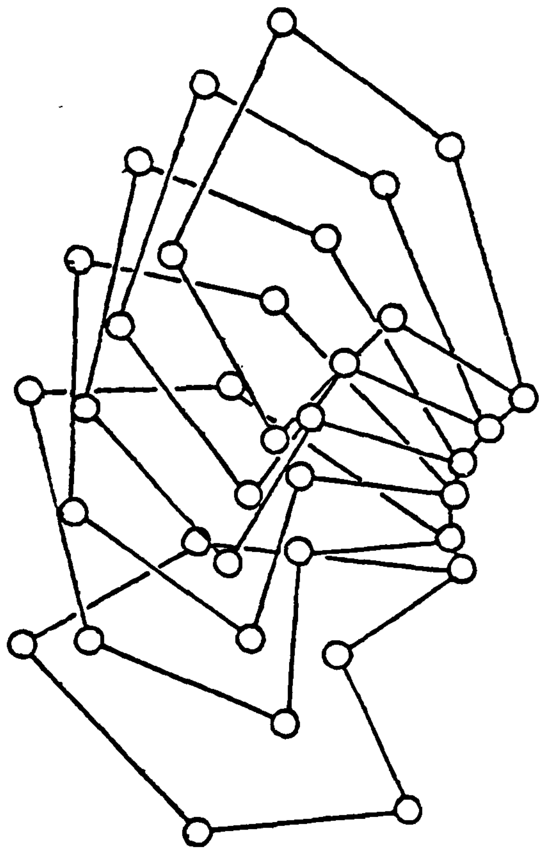


Figure 2-3: Tracking movement of an object using successive likelihood matches with a probabilistic graph model.



In this way, object model-based information may be incorporated using a priori probability structures. Many previous approaches to matching and pose estimation may be considered subsets of the probabilistic matching approach in which edge or region attributes of images are related by heuristic similarity measures rather than likelihoods.

2.3 Multiple Resolution Structural Basis Functions

Description of gray-scale shape in images is complex because shapes are often defined by some combination of region information and edge information. From the point of view of image processing, region information is often contained in the lower spatial frequency components of the image, while edge information is contained in the higher spatial frequency components. A complete description is difficult to achieve therefore from extraction of structural elements at one resolution level. A number of techniques have been proposed which transform the two-dimensional gray-level image to a three-space representation in (x,y,k) space, where k is the parameter of the resolution space. Such a representation has the advantage that peak structures in the (x,y,k) space shift uniformly along the k axis under scale transformations, and therefore objects of different size may be recognized in a representation with the same structural relationships. In this section we describe an extension of the multiple resolution tree which incorporates structural basis functions at each level in order to provide a more complete description at each level and include orientation specific structural components at each resolution level.

Our preliminary work on representation and probabilistic matching of multiple resolution structures has been carried out using a reversible transform called the "difference of low-pass" or DOLP transform developed by Crowley [Crowley 81, Crowley and Stern 84, Crowley and Parker 84]. The DOLP transform expands a single gray-level image f(x,y) into a set of bandpass images b(x,y,k), where k is the index of the multiple resolution tree. Each bandpass image is obtained by convolution of the original image with an appropriate bandpass impulse response function h(x,y,k). In the implementation of this transform by Crowley [Crowley 81], the bandpass impulse responses are constructed out of difference of Gaussian kernels. The most efficient implementation utilizes properties of the Gaussian kernel function which permits resampling and cascaded convolution with expansion and reduces the sequential computation of this transform from O(N..) multiplies and additions to O(N).

The DOLP transform set itself is not a very efficient representation of the image since it requires expanded storage space in a normal digital representation. However, it is possible to extract a symbolic representation of important information from the DOLP transform using peaks of the transform arrays as key structural elements of the image. In [Crowley and Sanderson 84], we introduced two levels of symbolic representation. The first level is composed of symbols derived directly from the DOLP images based on local positive maxima or negative minima in one, two, or three dimensions of the DOLP space representation. The second level of symbols utilize the connectivity among peaks and ridges to form peak paths and ridge paths. These symbol structures are defined in detail in [Crowley and Sanderson 84].

Figure 2.4 shows an example of a DOLP representation tree of peaks for the gray-level image of a bolt. The attributes indicated for each peak include the (x,y) position of the peak and the intensity of each peak. Figure 2.5 shows the resulting peak path representation for the same image. The connected peaks from figure 2.4 are reduced to individual symbols in figure 2.5. The set of attributes in the peak path model is expanded to include relative distances in the resolution space between maxima and number of children peak paths which descend. The resulting multiple resolution peak path tree is treated using the probabilistic matching methods described in the previous section. Outcome probabilities and attribute density functions are defined for the peak paths and likelihood-based search and correspondence matching are used to recognize and classify gray-level images of objects.

The advantage of the multiresolution representation techniques is the ability to describe both high resolution and low resolution structural image features in the same representation. The disadvantages of the bandpass filter approach are the difficulty in describing complex shapes, particularly those involving oriented components and the current demands of the computation to compute such extensive filtering operations. In order to enrich the capabilities of the multiple resolution transform, we have introduced a set of basis functions at each resolution level which includes oriented two-dimensional basis functions. This set of structural basis functions provides a much more complete description of the image at each level, at the expense of redundant information and increased computational load. In the context of this project we propose to explore the implementation of such techniques using parallel and optical processors. In this context the basis function tree provides a richer source of information for matching and interpretation which may be searched interactively rather than exhaustively computed.

Several options have been explored for the creation of an orthogonal basis set within the multiresolution framework. These strategies include:

- Linear combinations of Gaussian functions which form mutually orthogonal sets. This approach is a consistent expansion of the DOLP transform approach. However, the generality of the composite Gaussian approach is limited, and heuristic assumptions are required in order to provide an extensible orthogonal set.

Figure 2-4: DOLP representation tree of peaks for the gray-level image of a bolt.

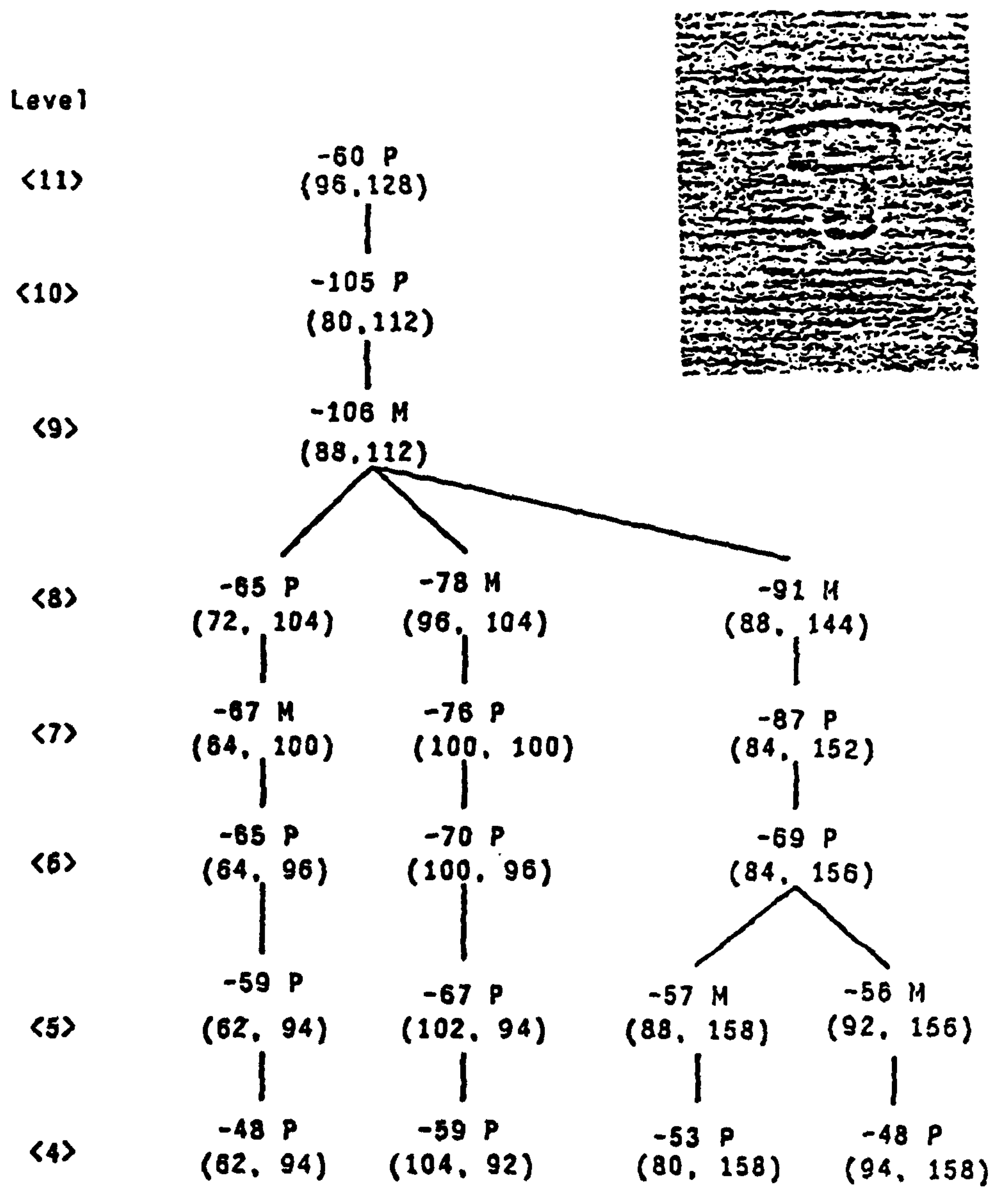
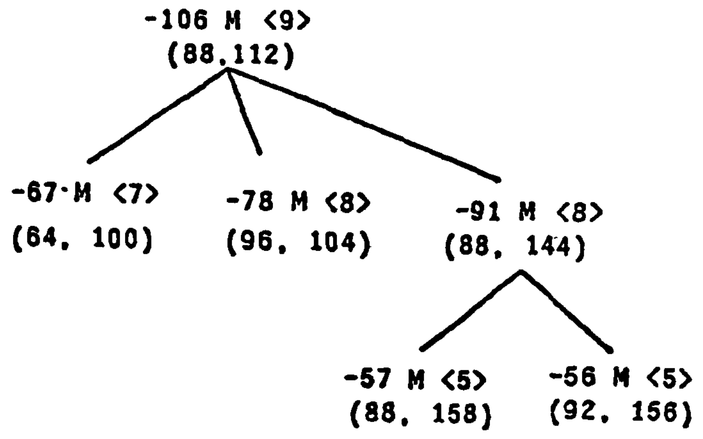


Figure 2-5: Peak path representation for the gray-level image of a bolt.



- Generalization of structural element masks for point, edge, and line functions. Such masks are utilized commonly in digital image preprocessing and implemented as 3x3 or 5x5 masks. The standard masks are not orthogonal functions, though they provide some of the properties desired in terms of structural element detection at various resolutions and orientation specific filters.

- Expansion of fundamental structural subpatterns of images. These subpatterns may include circular, rectangular, and elliptical components of successively smaller scale. This approach appears to be the most useful and incorporates important properties of the standard structural element masks in a more general form. In addition, the implementation of these basis functions utilizes a final smoothing window which reduces sidelobe leakage of the kernel functions and achieves some of the desirable localization properties of the Gaussian kernel.

Generation of the orthogonal basis functions described in the third strategy above may be based on a number of methods. One which appears to be useful here is the use of a relationship between the spatial domain kernel function and the frequency domain. As shown below, the mutually exclusive regions of the frequency spectrum guarantee orthogonality of the spatial kernel functions. This relationship can be exploited independently of the dimensionality of the function.

Given two band-limited signal vectors $\varphi_1(t)$, $\varphi_2(t)$ with non-overlapping frequency spectra $\Phi_1(\omega)$, $\Phi_2(\omega)$ such that

- $F_{1,2}(\omega) = \Phi_1(\omega)\, \Phi_2(\omega) = 0$, for all ω,

then taking inverse the inverse Fourier transform:

- $f_{12}(t) = \int_{-\infty}^{\infty} \Phi_1(t-\tau)\Phi_2(\tau)\, d\tau = 0,$

for all t. Thus if $f_{12}(t)$ is identically zero for all t, then it is zero for $t = 0$. If $\Phi_1(t)$ and $\Phi_2(t)$ are real functions, then

- $\int_{-\infty}^{+\infty} \Phi_1^*(t)\Phi_2(t)\, dt = \int_{-\infty}^{+\infty} \Phi_1(t)\Phi_2^*(t)\, dt = 0,$

and therefore $\Phi_1(t)$ and $\Phi_2(t)$ are orthogonal functions.

This formulation may be used to generate a set of mutually orthogonal functions which are expansions of rectangular and elliptical shapes in two dimensions. An example of such a one-dimensional set is:

- $\Phi_i(x) = \mathfrak{F}^{-1}[\, W_i(\omega) \sin(\omega/2)/(\omega/2)\,]$, where

- $W_i(\omega) = \mathrm{rect}(\omega/\pi - i\omega - 3\omega/2\,)$,

which provides an orthogonal basis set for expansion of the rectangular pulse of length 1. One such set $\{\Phi_i(x,k)\}$ may be derived at each resolution level specified by the length $k\Delta$. A related set may be derived in a similar fashion for the asymmetrical pulse yielding $\{\psi_i(x,k)\}$ which is orthogonal to $\{\Phi_i(x,k)\}$ at each level k. In practice, one may smooth these basis functions to reduce additional sidelobes. This final smoothing results in approximate orthogonality.

Extension of this approach to two-dimensions is achieved in a similar fashion and includes the partitioning of the frequency domain with respect to orientation as well. The resulting basis sets $\{\Phi(x,y,k), \Phi(x,y,k)\}$ form a multiresolution expansion at every point in the image. An observed brightness function f(x,y) may therefore be expanded around (x. ,y.) at resolution k as:

- $f(x,y) = \Sigma_{i=1} a_{ik} \Phi_i (x-x_0,y-y_0,k) + \Sigma_{i=1} b_{ik} \psi_i (x-x_0,y-y_0,k)$, where
- $a_{ik} = \iint_{-\infty}^{+\infty} f(x-x_0,y-y_0) \Phi_i (x-x_0, y-y_0,k)\, dx\, dy.$

The resulting multiresolution structure may then be searched for subpatterns expressed as sets of basis coefficients of the form $c = \{\ldots a_{ik} \ldots,\ldots b_{ik}\}$. Interactive search of this pattern space might be carried out using the types of hybrid process architectures discussed in this report. In addition, symbolic representation of these multiresolution expansions could be achieved using peak detection in the correlation space of any subpattern c.

2.4 Textural Surface Models

Texture occurs in images due to either irregular surface topography or to nonuniform surface reflectance. There have been a number of approaches to the modeling of texture in images [Haralick 70, Laws 79, Harwood et al 83]. Most of these rely on the modeling of local correlation properties of the gray-level image using either direct statistical measures or using the response to specific masks. In particular, [Laws 79] described a set of texture energy measures in terms of the response to nine linear 3 x 3 or 5 x 5 masks. These masks are chosen to reflect combinations of center-weighting, edge detection, and spot detection templates. The distribution of the outputs of these masks averaged across a textured regin was shown to be useful for the discrimination of texture types. Harwood [Harwood et al 83] extended this idea and studied the use of rank correlation statistics as a basis for discrimination.

Texture models such as those described above summarize descriptions of the variations in image intensity, but do not relate image properties to either surface topography or surface reflectance. An alternative model of image texture has been proposed using fractal geometries to model image texture and relate image texture to surface topography. Fractal geometry was introduced by [Mandelbrot 77, Mandelbrot 82] to describe certain classes of irregular edges or surfaces including coastlines and mountain profiles. More recently, [Pentland 83a, Pentland 83b] proposed the use of the fractal dimension to characterize images of natural scenes and perform texture segmentation.

The fractal dimension D is the dimension of a measurement space expressed relative to the topological dimension E. If the parameter $H = D - E$ is used to characterize the roughness of the observed texture, then $H = 0$ corresponds to a flat plane, while $H = 1$ corresponds to an array of spikes covering the plane. In terms of H, the cumulative distribution function of the fractal Brownian function B(t) is:

$$F(n) = Pr([B(t+\Delta t) - B(t)]/|\Delta t| < n),$$

where if $H = 1/2$ and if F(n) is a zero-mean, unit variance Gaussian, the B(t) defines a classical Brownian motion process. The fractal dimension in this sense is a compact parametric description of a homogeneous, isotropic random process which has advantages since it is invariant to scale. Pentland [Pentland 83a, Pentland 83b] related the image texture model to a natural surface model and showed that if the surface topology is fractal then the image intensity is also fractal if the surface obeys Lambertian assumptions. This result for fractal models is suggestive of more powerful results which might be achieved by relating image texture to more general random models of surface topography.

Studies of random surface topography suggest four principal contributions to the resulting textured image:

- Local edges of the surface elements
- Shading due to surface gradient and reflectance
- Shadows due to disparities between light incidence and viewing angles
- Local edges of one surface element occluding another.

These mechanisms are associated with surface topography and not with reflectance changes due, for example, to surface markings. The observed image texture varies in predictable ways with angle of view and lighting directions, and we would like to identify image texture measures which provide consistent measures of such changes. We are using two types of random surface model to investigate these measures:

- Smooth, continuous non-occluding Gaussian surface model,
- Piecewise continuous Poisson patch surface model.

Under the assumption of Lambertian surface properties, we are studying the relationship of texture energy measures, correlation measures, fractal dimension, and structural basis function expansions as measures of image texture and their ability to discriminate underlying surface texture. The extension of these techniques will be to provide segmentation of images on the basis of surface texture and describe time-varying characteristics of underlying surface topography.

2.5 Summary

The techniques studied have individually proven to be effective in scene analysis and image understanding. Recognition of objects may be accomplished through probabilistic matching and multi-resolution basis functions. Texture-based models can provide information about scene background. In high-altitude, time-varying imagery, the first two techniques offer information about accumulation, depletion or utilization of particular objects while the texture models help provide an understanding of terrain and land use.

References

[Crowley 81] J.L. Crowley.
A representation for visual information.
PhD thesis, Carnegie-Mellon University, Nov., 1981.

[Crowley and Parker 84]
J.L. Crowley and A.C. Parker.
A representation of shape based on peaks and ridges in the difference of low pass transform.
IEEE Trans. on PAMI, March, 1984.

[Crowley and Sanderson 84]
J. L. Crowley and A. C. Sanderson.
Multiple Resolution and Probabilistic Matching of 2-D Grey-Scale Shape.
In *Proceedings 2nd IEEE Computer Society Workshop on Computer Vision, Representation, and Control*, pages 95-105. IEEE, May, 1984.

[Crowley and Stern 84]
J.L. Crowley and R.M. Stern.
Fast computation of the difference of low-pass transform.
IEEE Trans on PAMI, March, 1984.

[Haralick 70] R.M. Haralick.
Statistical and structural approaches to texture.
Proc. IEEE 67:786-804, 1970.

[Harwood et al 83]
D. Harwood, M. Subbarao, and L.S. Davis.
Texture Classification by local rank correlation.
Technical Report TR-1314, University of Maryland Computer Science, August, 1983.

[Laws 79] K.I. Laws.
Texture Energy Measures.
In *Proc. Image Understanding Workshop*, pages pp 47-51. 1979.

[Mandelbrot 77] B.B. Mandelbrot.
Fractals, Form, Chance, and Dimension.
W.H. Freeman and Co., San Francisco, 1977.

[Mandelbrot 82] B.B. Mandelbrot.
The Fractal Geometry of Nature.
W.H. Freeman and Co., San Francisco, 1982.

[Pentland 83a] A. Pentland.
Fractal-based description of natural scenes.
Proc. IEEE CVPR :pp 201-209, July, 1983.

[Pentland 83b] A. Pentland.
Fractal Textures.
Proc. IJCAI 1983 :pp. 973-981, 1983.
Karlsruhe, Germany.

3. HIGH FRAME RATE SUB-PIXEL TARGET TIME-CHANGE STUDY

3.1 INTRODUCTION

As noted in Chapter 1, image registration is a key operation required in obtaining time-history data. Once successive frames or points in successive frames have been registered, the distortion function between the image frames (due to platform motion, etc.) can be determined and removed if necessary using SDFs. We can then integrate successive frames and subtract these frames to achieve a time-history track file on moving objects in the field-of-view of the sensor. This is the most significant bandwidth compression aspect of such processing, since only the changing information need be used.

As a particular case-study, let us consider the detection of a missile launch. In this case, we assume a high-resolution staring mosaic-sensor in geostationary orbit operating in several wavelength bands that is monitoring known missile locations (determining these locations is also possible with SDFs). The sensor system is assumed to have a high frame-rate and its resolution and altitude are such that the missile target is sub-pixel in size. To obtain accurate time-history information, we must thus register successive image frames to sub-pixel accuracies (0.01 of a pixel is typical) before subtracting the frames. When this has been done for many successive frames, a missile launch will emerge as a line or curve in the composite output difference image.

With this scenario and SBIU application defined, let us now discuss how to achieve sub-pixel image registration accurately using hybrid optical/digital techniques. We first recall (Table 1.1) the very important facts that we know that the two successive frames correlate and that the shift between frames is at most one pixel. We utilize this information in our new solution proposed below. The general technique is best described with respect to the simplified diagram in Figure 3.1. In this figure, we show the correlation of two successive frames of data to extract the amount (x_c, y_c) by which the second image must be shifted with respect to the first. We then shift and interpolate this second image frame and subtract it from the first frame. This produces time-history track files on candidate targets in the sensor's field-of-view. Artificial intelligence and image understanding techniques plus context information can be used on these time-history outputs (of the multiple targets expected) to arrive at a decision.

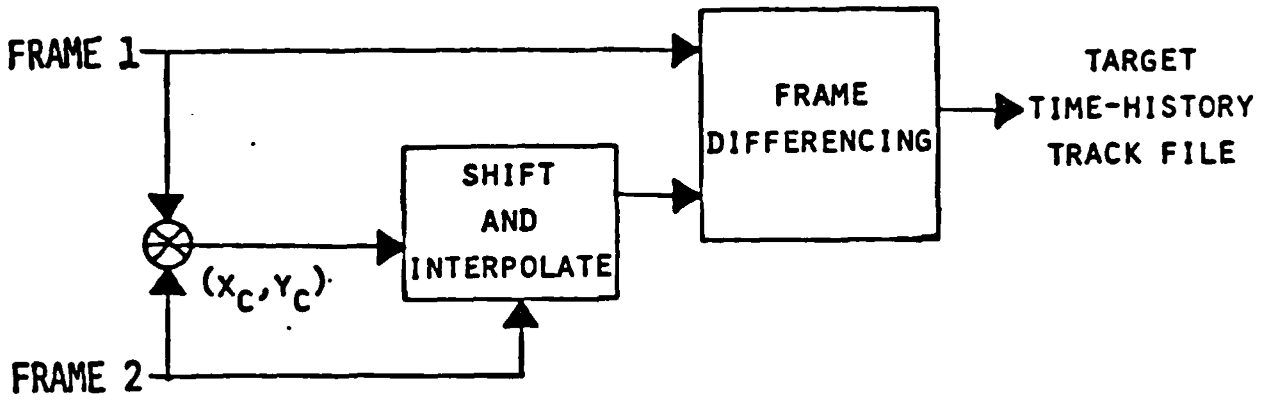


Figure 3-1: Simplified Hybrid Image Registration Concept

In our new proposed technique to achieve sub-pixel image registration, we will also utilize the fact that the shift between image frames is less than or equal to one pixel. Thus, we will place several detectors only around the center of the output correlation plane and are assured that we will have adequately sampled the output correlation. To obtain sub-pixel correlation peak location accurately, we feed these correlation plane sampled values to a digital estimator from which the sub-pixel shift τ estimation is obtained.

In our new system concept (Figure 3.2), we iterate the τ estimate until the registration accuracy is as small (Δ) as is desired. On each cycle, we re-interpolate and re-sample (i.e. sub-pixel shift) the image. Presently, the optical system performs the correlation. Methods to use the optical system to achieve interpolation will be addressed later.

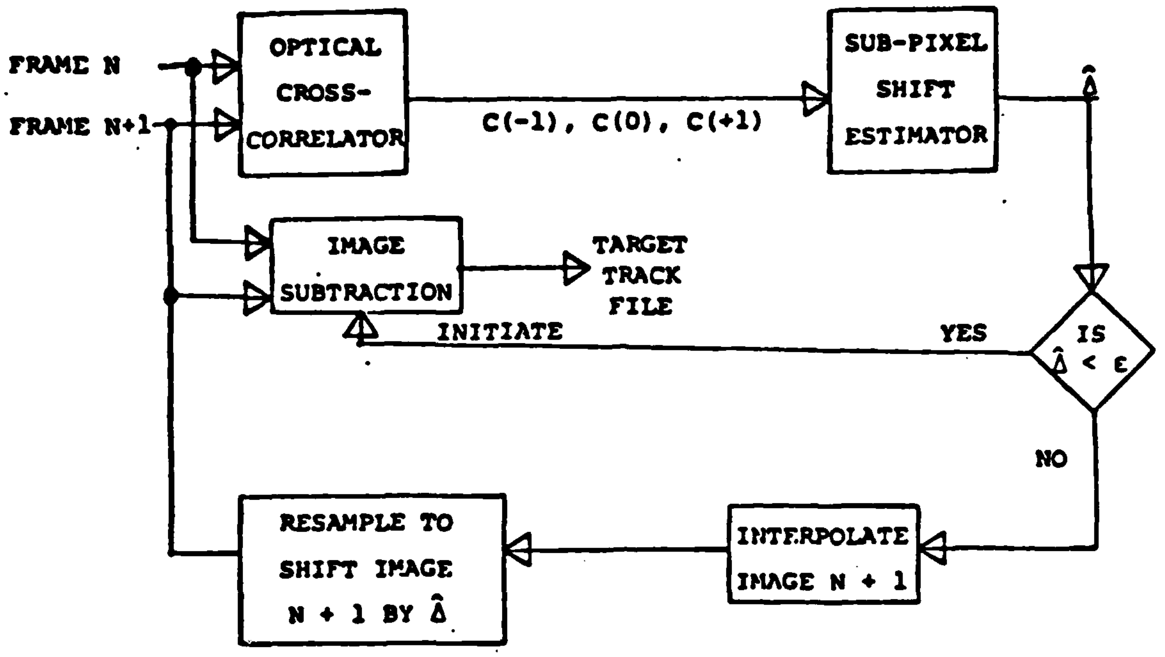


Figure 3-2: New Iterative Sub-Pixel Image Registration System

In Section 3.2, we discuss the space-based imagery we have produced for our initial tests. Three estimators for sub-pixel image shifts were investigated. These are described in Section 3.3. Our initial simulation results are advanced in Section 3.4 with attention to various sub-pixel estimators and their performance accuracy. Interpolation routines used are described in Section 3.5. The dynamic range and noise performance of the system are detailed in Section 3.6 and the performance of the system for target detection and tracking are presented in Section 3.7. This represents the first demonstration of sub-pixel target extraction, detection and tracking.

3.2 SPACE-BASED IMAGE DATABASE

To obtain quantitative data for analysis and demonstration, we employ digital techniques to produce simulated space-based imagery. The complete image scenario includes several layers of clouds with an earth background and a sub-pixel target. Each layer is known to be modeled in the IR wavelength band as a pseudonoise (PN) distribution with given correlation lengths, means and variances [Rauch et al, 1981]. Our initial tests considered only one layer with uniform statistics. Both correlated noise (CN) and uncorrelated noise (UCN) are included in the imagery. The CN corresponds to the cloud imagery in the IR and the UCN to the sensor and detector noise. The CN is coherent frame-to-frame (i.e., the same seed is used to generate the PN distribution in each frame). The UCN is non-coherent frame-to-frame (i.e., a different seed is used during PN generation).

The basic PN sequence generation techniques in IMSL [IMSL, Inc., 1982] produce random numbers, Gaussian-distributed with zero-mean and unit variance. Each new seed produces a different PN sequence. These sequences are white Gaussian noise with infinite bandwidth. To produce the correlated PN sequence needed for our image model, we must be able to control the correlation lengths and correlation coefficients ρ_x and ρ_y (horizontal and vertical) as well as the standard deviation (σ) of the sequence. A Markov exponential model was used for the autocorrelation function R(x,y) of the sequence

$$R(x,y) = \sigma^2 \rho_x^{|x|} \rho_y^{|y|}. \tag{3.1}$$

We wrote our own software for generating sequences with above Markov correlation function as IMSL does not provide this.

Let f(x,y) denote the UCN image of size 512 x 512 generated using IMSL. Then, we generate a

512 x 512 pixel UCN image f(x,y) starting at the upper left corner and proceeding to the bottom right corner, one row at a time. The pixel value $\hat{f}(x,y)$ at location (x,y) in the CN image is a function of the three neighboring $\hat{f}$ pixel values previously determined and the present UCN image value f(x,y). The four pixel values in question are shown below.

$$\begin{array}{|c|c|}\hline \hat{f}(x-1,y-1) & \hat{f}(x-1,y) \\ \hdashline \hat{f}(x,y-1) & \hat{f}(x,y) \\ \hline \end{array} \tag{3.2}$$

The equation by which the present $\hat{f}(x,y)$ pixel value is computed is

$$\begin{aligned} \hat{f}(x,y) = & -\rho_x \bullet \rho_y \bullet \hat{f}(x-1,y-1) + \rho_y \bullet \hat{f}(x,y-1) + \rho_x \bullet \hat{f}(x-1,y) \\ & + \sigma \bullet \{[1 - \rho_x{}^2 - \rho_y{}^2 + (\rho_x \bullet \rho_y)^2]^{0.5}\} \bullet f(x,y) \end{aligned} \tag{3.3}$$

This procedure converts an uncorrelated zero-mean image of variance σ^2 into a 2-D image of zero-mean and unit variance with correlation coefficients ρ_x,ρ_y. The correlation coefficient between diagonally adjacent elements is given by ρ_{xy}.

Ten versions of such a 2-D image were produced with intended correlation lengths $\rho_x = 0.875$ and $\rho_y = 0.925$. Values of ρ from 0.8 to 0.95 are typical of the values that represent cloud levels in the IR band being considered [Rauch et al, 1981]. Ten additional sets of 2-D images, with $\rho_x = \rho_y = 0.950$ were produced. These 20 images were then analyzed by correlation techniques. The measured mean, variance and correlation coefficients for these data are listed in Tables 3.1 and 3.2. As seen, the data exhibits very nearly the exact desired statistical parameters. The parameters are then estimated from the images generated as below.

$$\text{MEAN} = m = \sum_{x=1}^{512} \sum_{y=1}^{512} \hat{f}(x,y)/512^2$$

$$\text{VARIANCE} = \sum_{x=1}^{512} \sum_{y=1}^{512} [\hat{f}(x,y) - m]^2/512^2$$

$$\hat{\rho}_y = \sum_{x=1}^{512} \sum_{y=1}^{512} \hat{f}(x,y) \otimes \hat{f}(x,y+1)/512^2$$

$$\hat{\rho}_x = \sum_{x=1}^{512} \sum_{y=1}^{512} \hat{f}(x,y) \otimes \hat{f}(x+1,y)/512^2 \qquad (3.4)$$

In practice, the estimated means are subtracted from the data in the numerator in calculating ρ and a normalization using estimated variance is needed. But, in our case, the data is of zero-mean and unit variance and thus these corrections are not included in (3.4).

Table 3-1: Calculated statistical parameters for ten 2-D images of zero-mean and unit variance with $\rho_x = 0.875$ and $\rho_y = 0.925$

IMAGE NO.	MEAN	VARIANCE	ρ_x	ρ_y
1	-0.001	1.006	0.861	0.916
2	+0.060	1.018	0.865	0.916
3	0.025	1.038	0.865	0.918
4	0.056	1.027	0.863	0.917
5	0.039	0.966	0.859	0.915
6	0.002	1.025	0.865	0.918
7	-0.047	0.989	0.861	0.915
8	0.089	1.055	0.867	0.918
9	0.058	1.027	0.863	0.918
10	0.021	0.935	0.854	0.911

In Figure 3.3, we show the central regions of the autocorrelation functions of the CN images with $\rho_x = \rho_y = 0.1$, $\rho_x = \rho_y = 0.8$ and $\rho_x = \rho_y = 0.95$. For the Markov model autocorrelation function in (3.1), we see that R(x,y) decreases monotonically as |x| or |y| increases. The |x| and |y| values for which R(|x|,0) and R(0,|y|) equal 0.5 of the peak value are usually referred to as L_x and L_y, the correlation lengths. It can be easily seen from (3.1) that L_x and L_y are directly proportional to ρ_x and ρ_y. This can be easily seen from Figure 3.3, where we see that small ρ_x values indicate sharper correlation peaks. The observed correlation shape in any particular experiment can be used to estimate the underlying ρ_x,ρ_y values. This is made use of in deriving estimators for sub-pixel shifts in Section 3.3.

Table 3-2: Calculated statistical parameters for ten 2-D images of zero-mean and unit variance with $\rho_x = \rho_y = 0.950$

IMAGE NO.	MEAN	VARIANCE	ρ_x	ρ_y
1	0.001	0.993	0.929	0.929
2	0.121	1.033	0.933	0.931
3	0.048	1.045	0.931	0.931
4	0.118	1.075	0.932	0.933
5	0.081	0.931	0.926	0.928
6	0.006	1.029	0.931	0.932
7	-0.100	0.972	0.929	0.927
8	0.179	1.092	0.934	0.933
9	0.100	1.017	0.928	0.931
10	0.035	0.915	0.926	0.924

We add UCN, i.e. a standard IMSL PN sequence, to the 2-D images and embed a 4 x 4 pixel target in the 512 x 512 image. This completes our initial space-based image model. We next discuss how background and target shifts (sub-pixel) of data are generated. First, we consider the actual steering mosaic sensor detector array that we assume is used to produce the imagery to be processed. From prior analyses [Rauch et al, 1981], the targets of interest are known to be sub-pixel, specifically 0.25 of a detector pixel. We acheive this detector image and sub-pixel shifts by condensing each 8 x 8 pixel region of the original 512 x 512 image into one detector pixel. This should ideally produce an observed image of size 64 x 64. But since we plan to introduce shifts in the high resolution image, we cannot use the boundary points in the high resolution image. Then we ignore 4 rows or columns of pixels from each border and produce only a 63 x 63 size detector image. Sub-pixel shifts in the detector images can be obtained by using integer delays in the high

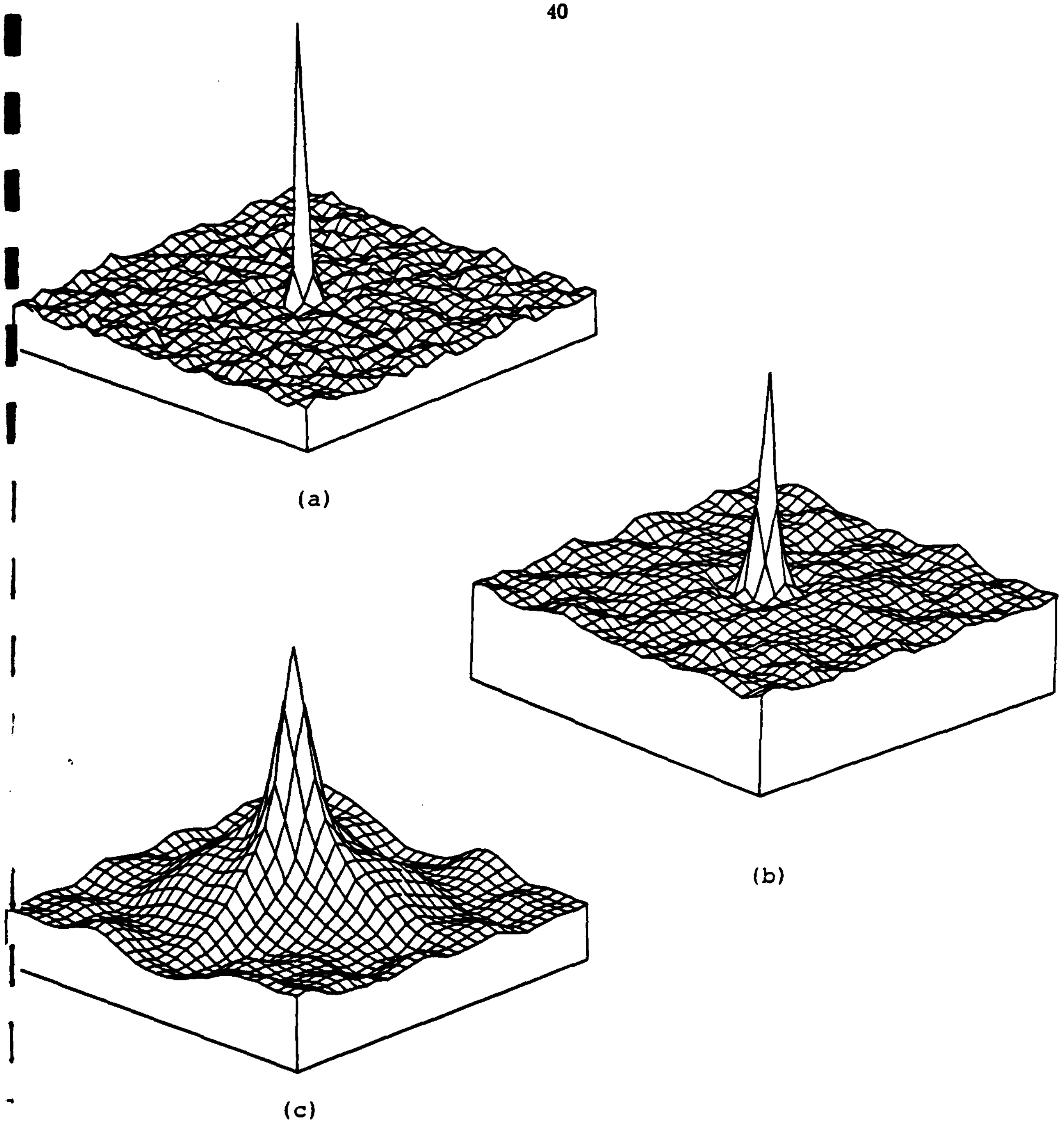


Figure 3-3: Correlation pattern for 2-D correlation sequences with: (a) $\rho_x = \rho_y = 0.1$, (b) $\rho_x = \rho_y = 0.8$ and (c) $\rho_x = \rho_y = 0.95$

resolution image followed by subsampling by a factor of 8. In addition to subsampling, we need to include the effects of detector blur function. The detector blur function we use is shown in Figure 3.4.

The detector blur function D(x) is based on the error function. It is unity in the center and essentially zero beyond 3σ. In 1-D it is defined as the integral of a Gaussian distributed function

$$D(x) = \begin{cases} 0 & x \geq x_0 \\ G(x) & x_1 \leq x \leq x_0 \\ 1 & 0 \leq x \leq x_1 \\ D(-x) & 0 < x \end{cases} \tag{3.5}$$

where $x_0 = 3\sigma + F/2$, $x_1 = -3\sigma + F/2$, F is the footprint size (8 x 8 pixels here, i.e. the image resolution in pixels), and the function G(x) is

$$G(x) = \frac{\int_a^b e^{-(z/\sigma)^2/2} dz}{\sqrt{2\pi}\ \sigma} \tag{3.6}$$

where $b = 3\sigma = 2$ pixels and $a = x - F/2 = x - 4$ pixels are the integration limits in (3.6). Eq. (3.5) indicates that the detector blur function D(x) is symmetric, has a flat central region and drops off to zero at the edges. The Gaussian shape is a good model for many of the sources of blurring such as atmosphere, imaging system, etc.

To produce sub-pixel image and target shifts, the original 512 x 512 scene is shifted, i.e. each 8 x

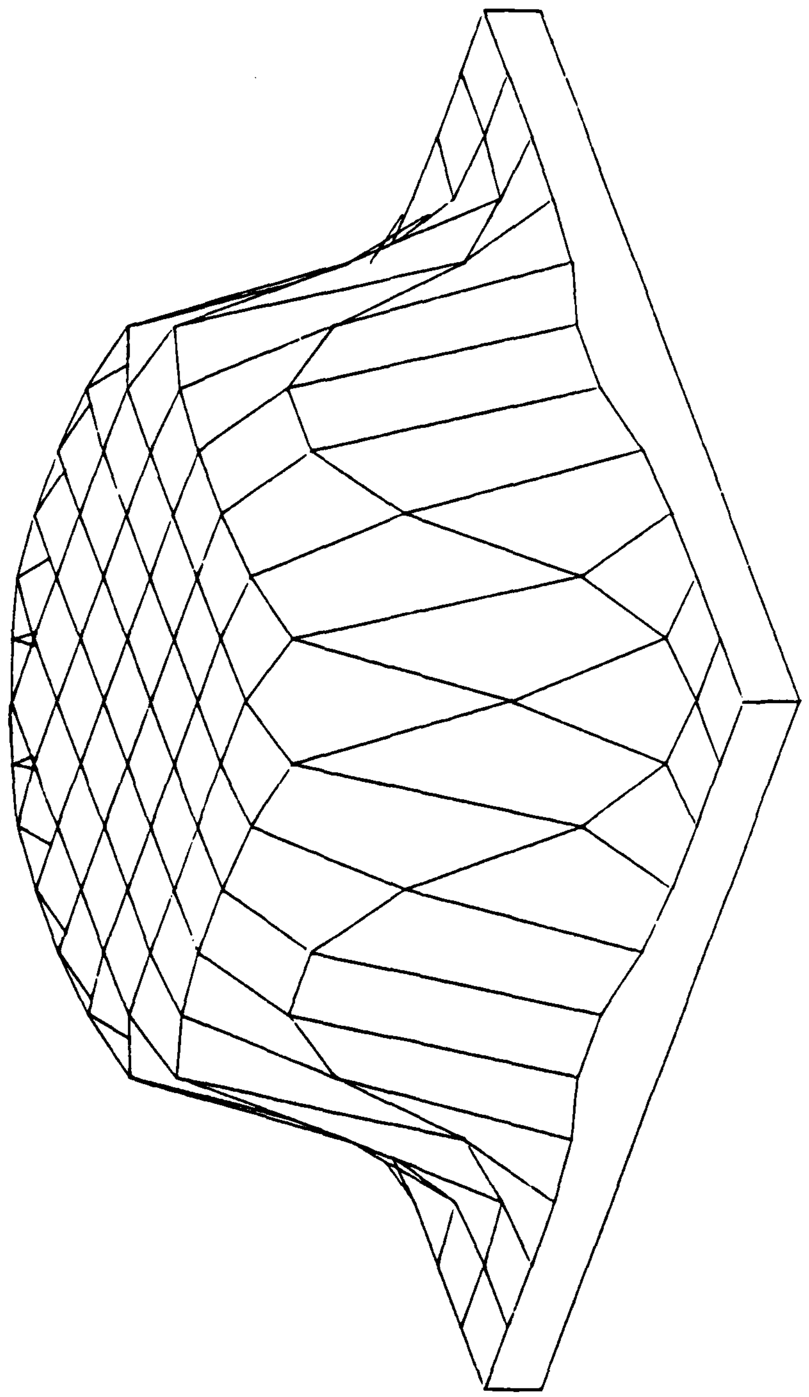


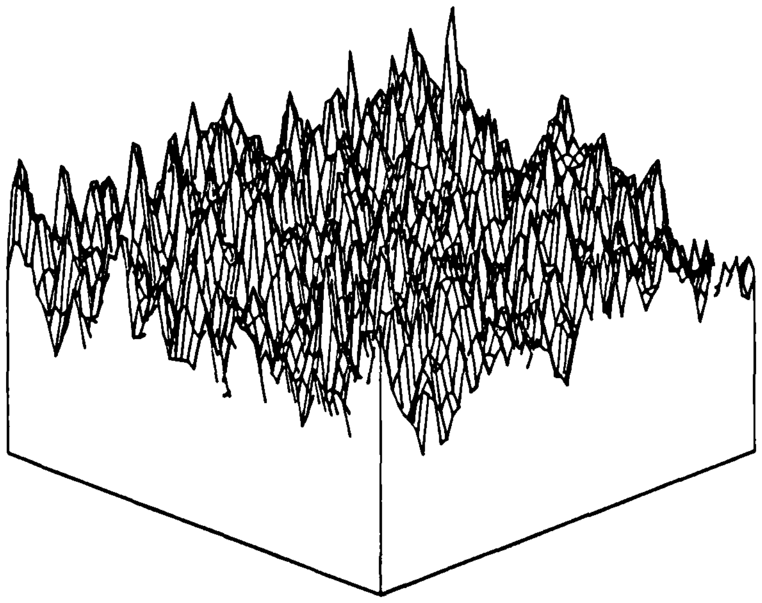
Figure 3-4: Detector blur function.

8 pixel region integrated to produce each detector value is shifted. A shift by one pixel in the original image is thus equivalent to 0.125 pixel shift in the detector image. The target is 4 x 4 pixels in the original scene and is thus 0.25 pixels in the detector image. UCN is added to produce the final image. This uncorrelated noise is non-coherent between successive frames. The background is shifted and so is the target. The target moves more than the background, as is expected. In Figure 3.5a, a typical CN and target image is shown. The CN in this image has $\rho_x = 0.95$ and $\rho_y = 0.95$. The target intensity is equal to that of the CN in this image. As can clearly be seen, no target is visible in this image. Figure 3.5b shows the same image with UCN also added. The strength of the UCN is 3dB below that of the CN. We cannot easily locate the target in either of the two images. Thus, the problem with detecting a sub-pixel target in such a space-based scene is clearly quite formidable.

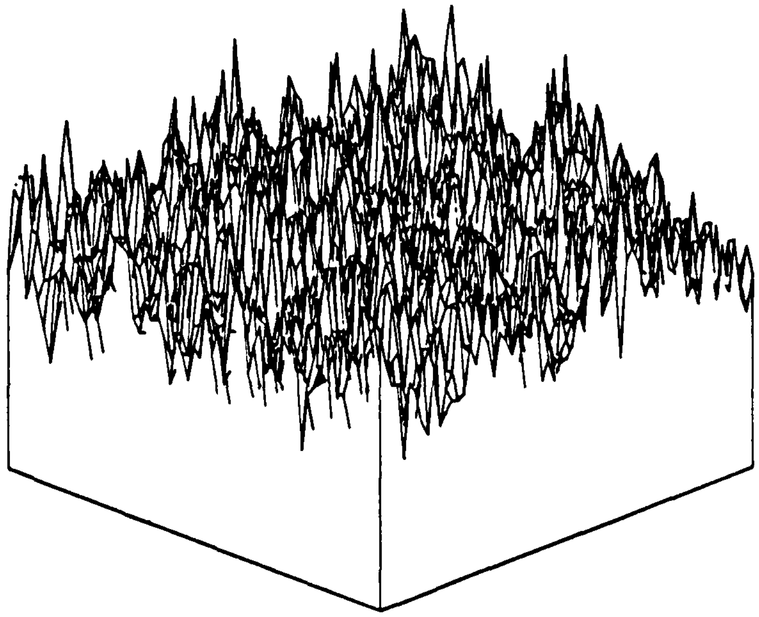
3.3 SUB-PIXEL ESTIMATORS

From Figure 3.3, we see that the shape of the correlation peak reflects the CN information in the scene. Because successive frames are taken quite close together in time (0.01 seconds), the CN information does not change appreciably. Thus, an output correlation between two successive frames is expected and the correlation peak is expected to be quite close to the center of the correlation plane, i.e. sub-pixel shifts between two frames are expected. To measure this shift, an estimator is used. The three estimators we considered estimate the shift τ from gradient estimates and from sampled correlation plane data. We denote the two successive image frames by $I_1(x,y)$ and $I_2(x,y)$. The CN or image background between the two images are shifted by (x_c,y_c). The estimates we obtain for these image shifts are denoted by $(\hat{x},\hat{y})$.

The correlation coefficient used in the simulation of high resolution images is 0.95. After the 8 x 8 pixel averaging, the low resolution detector image displays a correlation coefficient of 0.75; since the background in $I_1(x,y)$ and $I_2(x,y)$ (successive detector images) differs by less than a pixel, whereas the desired target moves by more than a pixel, we can extract the target by subtracting



(a)



(b)

Figure 3-5: (a) CN and target [$\rho_x = 0.95$, $\rho_y = 0.95$, equal target and CN intensity], (b) CN, target, and UCN [UCN is 3dB below CN in intensity]

$I_1(x,y)$ from $I_2(x,y)$. In the ideal case when the background does not change from $I_1(x,y)$ to $I_2(x,y)$, subtraction removes the background completely. But in practice, we can estimate by how much the background drifted between the two frames, compensate for it and then subtract the two frames. This process will distort the target, but since many successive frames will be used to form target track, this will have little effect.

3.3.1 Gradient-Based Estimator [Lucas and Kanade, 1981]

We will consider 1-D notation for simplicity. The two observed images $I_1(x)$ and $I_2(x)$ are related by a sub-pixel shift, namely Δx, i.e.,

$$I_2(x) = I_1(x-\Delta x), \tag{3.7}$$

where we ignored the UCN for the moment. We know that the spatial derivative of the image $I_1(x)$ can be approximated as

$$I'_1(x) = \frac{dI_1(x)}{dx} \simeq \frac{I_1(x) - I_1(x-\Delta x)}{\Delta x} = \frac{I_1(x) - I_2(x)}{\Delta x}, \tag{3.8}$$

where $I'_1(x)$ denotes the spatial derivative of $I_1(x)$. The approximation used in (3.8) is useful only if Δx is small, i.e., only if we are concerned with sub-pixel shifts. From (3.8), the sub-pixel shift can be estimated as

$$\hat{\Delta} x = \frac{I_1(x) - I_2(x)}{I'_1(x)}. \tag{3.9}$$

However, in this method a new quantity can be computed for the right-hand side of (3.9) for each spatial position x measured. Thus, we can improve the estimate for $\hat{\Delta}x$ by averaging the right-hand side of (3.9) over all possible x values in the images $I_1(x)$ and $I_2(x)$. But, then we have to be concerned with the possibility of $I'_1(x)$ being zero at a particular x value. Such a situation can lead to $\hat{\Delta}x$ being infinite. To avoid this, we introduce a weighting function w(x) and obtain the sub-pixel shift estimator equation

$$\hat{\Delta} x = \frac{\Sigma_x\{w(x)[I_1(x)-I_2(x)]/I'_1(x)\}}{\Sigma_x w(x)}, \tag{3.10}$$

where the summations are over all image pixels x and where

$$w(x) = \frac{1}{|I_2'(x) - I_1'(x)|}. \tag{3.11}$$

This weighting function is chosen to avoid those x values for which $I_1'(x)$ is zero. The weighting function used in (3.11) is inversely proportional to $I''(x)$, the second spatial derivative of I(x). Then this weighting function becomes small when there are large changes in the first spatial derivative I_1' in a small region.

To improve the estimation procedure, we use (3.10) iteratively. Initially, the two frames $I_1(x)$ and $I_2(x)$ are assumed to be in registration, i.e., $\Delta x = 0$; (3.10) is then used to estimate Δx between $I_1(x)$ and $I_2(x)$. This estimate is then used along with a proper interpolation routine to shift $I_1(x)$ by Δx to bring it into registration with $I_2(x)$. The procedure in (3.10) is once again repeated to get a better estimate for Δx. The iterative method thus follows the formula

$$\hat{\Delta}x(k+1) = \hat{\Delta}x(k) + \frac{\Sigma_x \, \{w(x)\,[I_1(x-\hat{\Delta}x(k)) - I_2(x)] \,/\, I'_1(x-\hat{\Delta}x(k))\}}{\Sigma_x \, w(x)} \tag{3.12}$$

where k denotes the iteration number and $\hat{\Delta}(0) = 0$.

The success of the iterative procedure in (3.12) depends mainly on the accuracy of the interpolator used to produce $I_1(x-\hat{\Delta}\,(k))$ at the k-th iteration. Good interpolators are very time consuming and thus *this procedure is time consuming*. Even though we have chosen the weighting function w(x) in (3.11), there are no fixed rules for optimal weighting functions. This last topic should be investigated in our future research.

In our experiments (Chapter 4), we have found that ten iterations in general are needed. Because of the interpolation needed, this approach is less attractive compared to the correlation plane methods. In general, producing the correlation function can be time consuming. However, this operation is easy to implement optically and since we also do not need to produce the full correlation plane, the detector requirements are also quite modest. Thus, this scenario, architecture and algorithm are quite practical if realized optically.

3.3.2 Exponential Correlation Estimator

In both of our correlation-based estimators, a shape is assumed for the correlation pattern. The correlation pattern between I_1 and I_2 is sampled at several pixel locations and from these correlation coefficient plane samples, the location of the peak is estimated to sub-pixel accuracy. In the exponential-correlation estimator, a Markov correlation function is assumed for the cross-correlation function c(x,y) of the two images $I_1(x,y)$ and $I_2(x,y)$.

$$c(x,y) = c_p \rho_x^{|x-x_c|} \rho_y^{|y-y_c|} \tag{3.13}$$

where c_p is the peak correlation value. The sub-pixel shifts x_c and y_c are between -0.5 and 0.5. This approximation to c(x,y) exactly matches the exponential CN sequence produced and is thus expected to yield quite excellent performance. Surprisingly, no prior use of such an estimator has been located in the literature. Since Markov type correlations are found in many aerial scenes, this should be useful in other applications also. To determine estimates $(\hat{x},\hat{y})$ for the sub-pixel shifts (x_c,y_c), we substitute six sampled c(x,y) values c(-1,0), c(1,0), c(2,0), c(0,-1), c(0,1) and c(0,2) into (3.13).

If we take the natural log of both sides of (3.13), we obtain

$$\ln[c(x,y)] = \ln[c_{peak}] + |x-x_c|\ln\rho_x + |y-y_c|\ln\rho_y. \tag{3.14}$$

Because of the absolute value signs on $(x-x_c)$ and $(y-y_c)$, the correlation plane value at the origin, namely c(0,0) is ambiguous concerning the signs of x_c and y_c. But, since x_c and y_c are guaranteed to have magnitudes less than 1/2, the other correlation plane values uniquely define all parameters. The unknown parameters on the right-hand side of (3.14) are c_{peak}, x_c, y_c, ρ_x and ρ_y.

We are interested in x_c and y_c only. To obtain $\hat{x}_c$, we eliminate the dependence on ρ_y by using

$$\hat{x}_c = \frac{\ln\{c(-1,0) / c(1,0)\}}{2\ln\{c(2,0) / c(1,0)\}} \tag{3.15}$$

From (3.14) for the no noise case, we find

$$\hat{x}_c = \frac{|-1-x_c| ln\rho_x - |1-x_c| ln\rho_x\}}{2\{|2-x_c| ln\rho_x - |1-x_c| ln\rho_x\}}$$

$$= \frac{\{+1+x_c-1+x_c\}}{2\{2-x_c-1+x_c\}} = x_c. \tag{3.16}$$

Similarly, we obtain $\hat{y}_c$ by eliminating the dependence on ρ_x as below.

$$\hat{y}_c = \frac{ln\{c(0,-1) / c(0,1)\}}{2ln\{c(0,2) / c(0,1)\}}. \tag{3.17}$$

For the no noise case, we can show that $\hat{y}_c = y_c$. Becaue of the nonlinear nature of the equations involved, there is no direct or unique method of deriving $\hat{x}_c$ and $\hat{y}_c$. Our estimators in (3.15) and (3.17) are attractive as they use only six values and use only simple *ln* operations. The statistical behavior of these estimates will be investigated in our future research.

It is possible to iterate this procedure (Section 3.., .o achieve successively better estimates. We will quantify the performance of this estimator in Section 3.4.

3.3.3 Parabolic-Correlation Estimator [Kumar et al, 1982]

The use of fitting a parabolic curve to the correlation peak when the delay is not an integer multiple of the sampling period is widely practiced [Boucher and Hassab, 1981]. In this case, the central region of the correlation peak is described by

$$c(x,y) = a + bx + cy + dx^2 + ey^2. \tag{3.18}$$

Evaluating (3.18) for different correlation plane sampled pixel values, we find the coefficients in (3.18) to be [Kumar et al, 1982]

$$\begin{aligned} a &= c(0,0),\ b = [c(1,0) - c(-1,0)]/2 \\ c &= [c(0,1) - c(0,-1)]/2 \\ d &= [c(1,0) + c(-1,0) - 2c(0,0)]/2 \\ e &= [c(0,1) + c(0,-1) - 2c(0,0)]/2. \end{aligned} \tag{3.19}$$

Setting the derivatives of (3.19) with respect to x and y equal to zero, the sub-pixel shift estimates are found to be

$$\hat{x}_c = - \frac{[c(1,0) - c(-1,0)]}{2\{2c(0,0) - c(1,0) - c(-1,0)\}}$$

$$\hat{y}_c = - \frac{[c(0,1) - c(0,-1)]}{2\{2c(0,0) - c(0,1) - c(0,-1)\}} \tag{3.20}$$

The estimators for sub-pixel shifts x_c and y_c are presented in (3.20). The correlation plane samples c(0,0), c(0,1), c(0,-1) and c(-1,0) are obtained by sampling the cross-correlation of the two images $I_1(x,y)$ and $I_2(x,y)$. We have shown [Kumar et al, 1982] that the use of the estimator in (3.20) leads to biased estimates of x_c and y_c for Markov-type images. This is one of the reasons why we considered the exponential model based estimator of the previous section. The popularity of parabolic estimator is one compelling reason for its use. It is also easier to implement as it does not involve log operations.

3.3.4 Least-Mean-Squared Error Estimator

The estimators suggested in (3.15), (3.17) and (3.20) make use of only a few (5 or 6) points in the central region of the cross-correlation between I_1 and I_2 to estimate x_c and y_c. If the cross-correlation operation is carried out digitally, use of only 5 or 6 points simplifies the computation involved. But optical correlators can produce the complete correlation plane without any additional effort. Thus, we consider the use of more information from the correlation plane to determine x_c and y_c. We can treat this as a curve fitting problem (parabola or exponential) to the observed data points.

Let us assume that we are using a matrix of n x n observed correlation plane values and that we attempt to model them by a second-order polynomial as in (3.18). Let us denote the five unknown parameters in (3.18), namely a, b, c, d, e, by a column vector $\underline{\theta}$ of size 5. The observed correlation data values can be represented in a lexicographic order as the column vector $\underline{c}$ of size n^2. Then $\underline{c}$ is related to $\underline{\theta}$ by

$$\underline{c} = \underline{A\theta} \tag{3.21}$$

where $\underline{A}$ is a matrix with n^2 rows and 5 columns. If the i-th element of $\underline{c}$ corresponds to the location (x,y) in the correlation plane, then the i-th row of $\underline{A}$ is given by

$$[1 \;\; x \;\; y \;\; x^2 \;\; y^2]. \tag{3.22}$$

In general θ can be obtained by solving (3.21), but we note that there are more equations (n^2) than unknowns (5). Thus, a solution for $\underline{\theta}^*$ is found only such that the square error between the observed correlation data $\underline{c}$ and $\underline{A\theta}^*$ is minimized. It is straightforward [Duda and Hart, 1973] to show that the solution is

$$\underline{\theta}^* = \underline{A}^+\underline{c}, \tag{3.23}$$

where $\underline{A}^+$ is the generalized inverse of $\underline{A}$.

This approach allows the use of an arbitrarily large number of samples in the correlation plane. But one has to carefully consider the ill-conditioning of $\underline{A}$ as more points are added. This occurs because the condition number increases as the matrix size increases. The condition number for the central 3 x 3 region in the correlation plane is 1799 while the condition number of the central 5 x 5 region is 15846. Thus fewer data points are preferred from the computational, accuracy and dynamic range considerations.

Even though we introduced our least mean squares error (LMSE) method using the five-point second-order polynomial in (3.18), we can use other models as well. Often, we may model c(x,y) as

$$\begin{aligned} c(x,y) &= \{\text{parabola in } x\}\{\text{parabola in } y\} \\ &= a + bx + cy + dxy + ex^2 + fy^2. \end{aligned} \tag{3.24}$$

The same approach as before can be used except that $\underline{\theta}$ will be a vector of size 6 and $\underline{A}$ will be a matrix with 6 columns. We can extend this technique to the exponential model also. But the equations resulting from minimizing the square error in general will be nonlinear. Linearization of such nonlinear methods will be considered in the future.

3.4 ESTIMATOR ACCURACY AND PERFORMANCE

In this section, we discuss our performance measures used and provide comparative tests on the quantitative performance of the estimators suggested in Section 3.3.

As our performance measure, we consider the sub-pixel estimation accuracy obtained and use this to select the estimator to use. Another useful performance measure is the mean-square value of the difference between the two image functions (after sub-pixel shifting one image by the amount calculated). This second performance measure combines measures for the sub-pixel shift estimator and the interpolator used to perform the sub-pixel image shifts. This second performance measure will thus be most useful in evaluation of the various interpolation routines we will employ. This issue is discussed in Section 3.5.

In Table 3.3, we show the sub-pixel estimates obtained after different numbers of iterations for the gradient-based estimator of Section 3.3.1. In this case, the data had $\rho_x = \rho_y = 0.950$, the sub-pixel shift was -0.25 in both directions, and only CN was present. Only data for a cubic spline interpolator is included. As seen, 10-15 iterations are necessary before the iterative estimator was close to the correct value. The same behavior was observed when a parabolic interpolator was used. Thus, we have chosen not to pursue this estimator in our further studies, since it requires a significant number of iterations, much more than do the other estimators. In the gradient algorithm tests, all image points were used as this should provide the best estimate. A detailed analysis of the estimation outputs from this gradient algorithm showed divergence for a linear interpolator. This and other aspects of this algorithm merit further attention. However, the fact that it requires more iterations than other methods makes it less worthy of further detailed study.

Table 3-3: Sub-pixel shift estimates for gradient-based estimator

ITERATION NUMBER	1	5	10	15
$\hat{x}$ estimate	-0.049	-0.162	-0.224	-0.247
$\hat{y}$ estimate	-0.052	-0.175	-0.251	-0.294

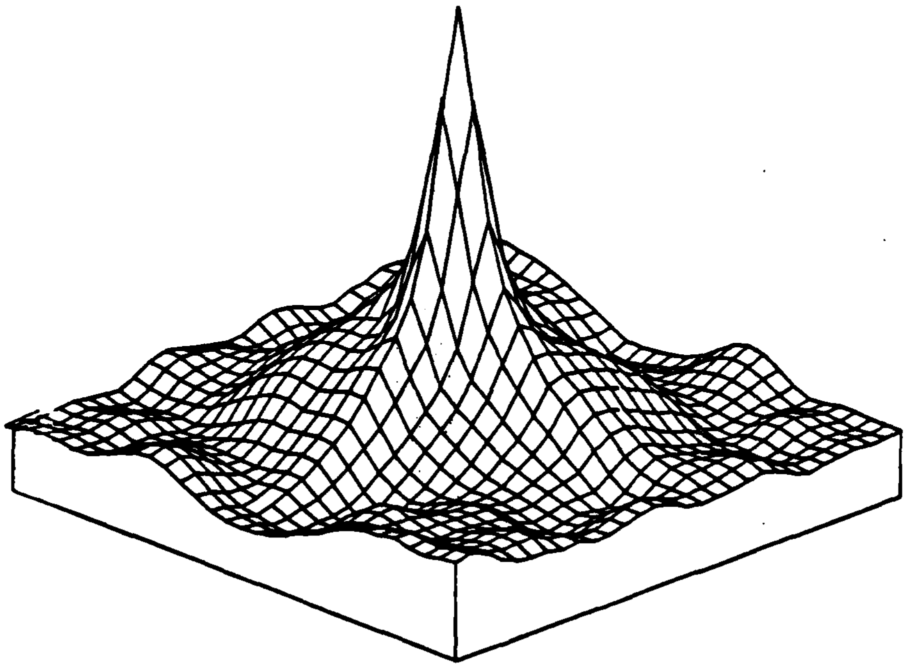
In Table 3.4, we show the sub-pixel estimates obtained after the first iteration for the exponential, parabolic and LMS 5-3 (5 coefficients and a 3 x 3 sample size in the correlation plane) correlation plane estimators. For these cases, only CN was present with $\rho_x = \rho_y = 0.95$ and the correct sub-pixel shift was $\hat{x} = \hat{y} = -0.25$ of a pixel in the detector image (two pixels in the original 512 x 512 image). As seen, the exponential estimator performs best. This is expected since the CN used had an exponential correlation function. The parabolic estimator performed quite well after only one iteration, as did the LMS estimator shown. The LMS estimator was evaluated for other cases (with more coefficients and with a larger correlation plane sampling region). In all of these cases, the LMS estimator performed worse. We feel that this may possibly be due to the larger condition number of the matrix in such problems.

Table 3-4: Sub-pixel shift estimates obtained with different correlation plane estimators

ESTIMATOR	EXPONENTIAL MODEL	PARABOLIC MODEL (5 COEFFS.)	LEAST MEAN SQUARE (5 COEFFS.)
$\hat{x}$ estimate	-0.2555	-0.1843	-0.1839
$\hat{y}$ estimate	-0.2401	-0.1844	-0.1868

A representative example of the autocorrelation of CN only image data with $\rho = 0.95$ was shown in Figure 3.3c and is repeated in Figure 3.6a for convenience. The cross-correlation for two such CN sequences with shifts of 0.375 pixels in x and 0.5 pixels in y is shown in Figure 3.6b. From these two figures, we see how the shape of the correlation function changes with sub-pixel shifts and that the accuracy with which the correlation peak can be measured directly (without the estimator) is quite poor.

In our future work, we will continue to use the exponential correlation estimator, since it performs so well and matches the exact correlation shape for the data presently being used. The parabolic estimator will also be retained, since it is reported to be more robust. It may perform better than the exponential estimator in the case of Gaussian noise and other natural image distributions that are not so well controlled statistically. The condition number of the correlation matrix in the LMS data should be investigated to determine if this is the reason for the reduced performance of this algorithm. In addition, the maximum likelihood and MAP estimators will be


(a)

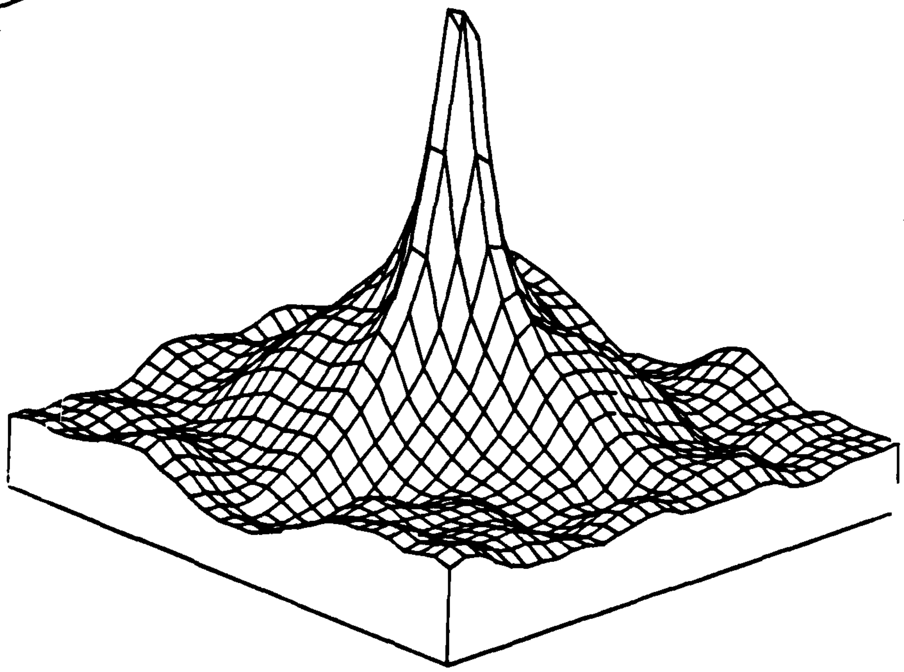

(b)

Figure 3-6: Correlation plane functions for CN with $\rho = 0.95$. (a) Autocorrelation, (b) Cross-correlation of the images with shifts of 0.375 pixels in x and 0.5 in y.

investigated as parametric estimators. The results obtained using these estimators will be compared to those containing the present non-parametric estimator. Our present performance has demonstrated the excellent sub-pixel accuracy possible after only one iteration in our estimators.

3.5 INTERPOLATOR PERFORMANCE AND SELECTION

Once the sub-pixel estimator has been obtained, the new image must be sub-pixel shifted by this amount (before being subtracted from the previous image). This requires interpolation. The three interpolation techniques we investigated (linear, second-order and cubic spline) are discussed in this section. Our results achieved using them are advanced and quantified. In these tests, as our performance measure, we use the mean square error of the difference between the two images after sub-pixel interpolation. To test only the interpolation algorithms, we apply the exact sub-pixel shift to each interpolator and then compute the mean square error of the two new images.

Let $I_1(x,y)$ and $I_2(x,y)$ be the two images being considered. The similarlity between the two images can be characterized by

$$\text{NMSE} = \frac{\Sigma_{x,y}(I_1(x,y) - I_2(x,y))^2}{\Sigma\, I_1^2(x,y)}, \tag{3.25}$$

where NMSE denotes normalized mean square error and where $I_1(x,y)$ is assumed to be of zero-mean. When $I_1(x,y)$ and $I_2(x,y)$ are identical, NMSE is seen to be zero. Thus smaller NMSE values indicate better sub-pixel shift estimation and interpolation.

The NMSE obtained using various interpolation given the exact sub-pixel shift between the two images is shown in Table 3.5.

A sub-pixel shift of 0.25 was used to obtain the above results. We see from the results in Table

Table 3-5: NMSE of the Interpolators

Linear	Parabolic	Spline(4)	Spline(5)	Spline(6)	Spline(7)
0.7675	0.0320	0.0357	0.267	0.0282	0.0289

3.5 that the linear interpolator performs very poorly and is not considered further. Spline(n) indicates n-th order spline interpolation. Fifth-order spline interpolation has the smallest NMSE, but parabolic has only 20% more NMSE. Since parabolic interpolation is computationally more efficient compared to spline(5), we will consider only parabolic for the future.

3.6 DYNAMIC RANGE AND NOISE CONSIDERATIONS

We have so far considered the behavior of different sub-pixel shift estimators under ideal conditions. We now investigate their accuracies in the presence of detector dynamic range limitations and detector noises.

3.6.1 Detector Dynamic Range

Varous points in the central region of the cross-correlation between the two frames need to be detected before the sub-pixel shift between the two frames can be estimated. Since we are concerned with the central 5 x 5 correlation plane region, we find the minimum and maximum magnitudes in these 25 points in the correlation plane. For a typical cross-correlation, we observed a minimum of 485 and a maximum of 2750 indicating a dynamic range requirement of about 6 or about 16dB. One can easily achieve a dynamic range of 1000 or 60dB with present detectors. Thus detector dynamic range should not pose any significant problems in sub-pixel shift estimation.

3.6.2 Input Noise

The next question to be investigated is the effect of input UCN noise on sub-pixel shift estimation. To simulate the detector noise, we add UCN noise with a prescribed variance to the input plane data. We show the estimated x-shifts and y-shifts for various amounts of input noise using both the parabolic and exponential estimators in Table 3.6.

Table 3-6: Estimated $\hat{\Delta x}$ and $\hat{\Delta y}$ under varying input noise. The true Δx and Δy are -0.25 and -0.25

INPUT SNR	$\hat{\Delta x}$	$\hat{\Delta y}$	$\hat{\Delta x}$	$\hat{\Delta y}$
$-\infty$	-0.18431	-0.18444	-0.25552	-0.24013
1000	-0.18431	-0.18444	-0.25552	-0.24013
100	-0.18431	-0.18444	-0.25552	-0.24012
10	-0.18431	-0.18444	-0.25550	-0.24011
5	-0.18433	-0.18444	-0.25556	-0.24011
4	-0.18432	-0.18446	-0.25552	-0.24001
1	-0.18426	-0.18437	-0.25553	-0.24003

We see from Table 3.6 that the sub-pixel shift estimates are very insensitive to input plane noise. Even with an SNR of only 1, the difference from the no noise case is only 0.00005. Thus, input noise seems to pose no problems in our method. This is a significantly advantageous feature of this sub-pixel shift estimtaion algorithm. In our next tests, we will add noise to the detector plane data.

3.7 TARGET DETECTION AND TRACKING

In this section, we present the results of target detection and tracking on the two image frames presented in Figures 3.5a and 3.5b. Figure 3.5a shows a 3-D display of the image of a target in background. The background is simulated by a CN image of size 63 x 63 as before. The correlation coefficients used were $\rho_x = 0.95$ and $\rho_y = 0.95$. The target is of sub-pixel size in the 63 x 63 image (0.5 of a pixel). The strengths of the target and CN are equal. We cannot identify the target from Figure 3.5a.

In Figure 3.5b, we have a 3-D display of the next simulated sequential image of this same scene. Between these two images, the target shifted by more than one pixel and the background by less than one pixel. As seen, figures in 3.5a and 3.5b differ only slightly. The CN which simulates the background moved by 1/4 of a pixel between the two frames. This movement was along the n-direction. The target moved 2 pixels in the x-direction and 1 pixel in y-direction between the two frames. The second frame also differs from the first in that there is UCN of half the CN strength in it. We cannot identify the target in either of Figures 3.5.

The results of our processing are shown in Figure 3.7. We cross-correlated the two images I_1 and I_2 in Figure 3.5 to estimate the sub-pixel shift of the background. We first subtract I_1 and I_2 without taking into account the background shift. We show the thresholded image of $|I_2-I_1|$ in Figure 3.7a. A thresholding at half the target strengths is used to provide convenient display. Multiple peaks are present in this causing confusion about the correct target location. I'_1 denotes the image obtained from I_1 by using the sub-pixel shift estimated from I_1 and I_2. The thresholded difference image $|I'_1$-I-'2'| shown in Figure 3.7b displays only two peaks near the center. These correspond to the location of the target in frames 1 and 2. Then all the background has been suppressed while the target movement is clearly shown. Similarly, uncompensated images I_2 and I_3 yield multiple peaks in their thresholded difference image, shown in Figure 3.7c. After proper shifting, the difference of I_2 and I_3 yield the thresholded difference image in Figure 3.7d. Once again the background has been completely eliminated. These results show the importance of sub-pixel shift estimation for background elimination and target extracction.

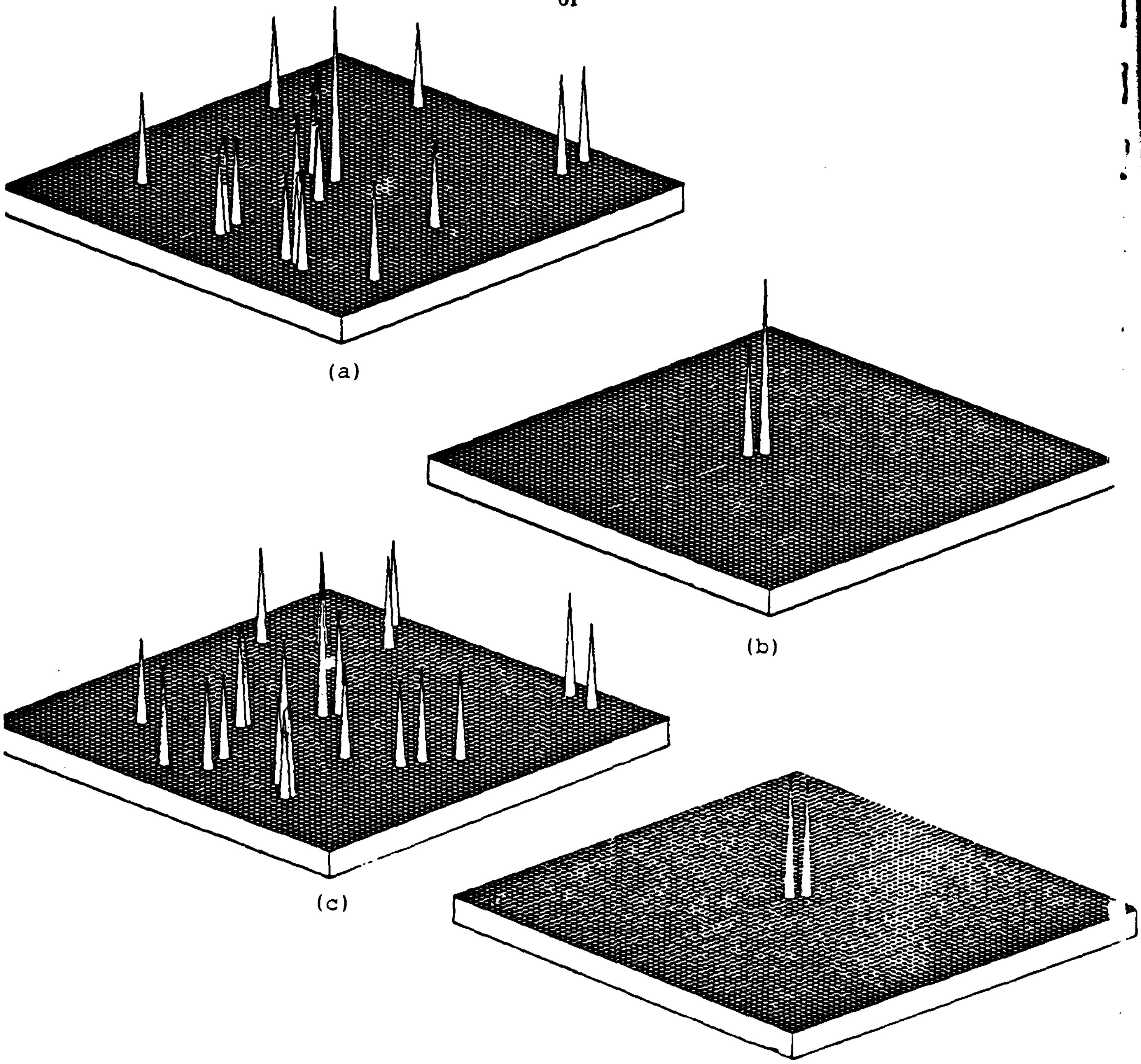


Figure 3-7: Thresholded differences (a) of the images in Figure 3.5; (b) in (a) after sub-pixel shift estimation using parabolic estimator and interpolator; (c) of frames 2 and 3; (d) in (c) after sub-pixel shift parabolic estimation and interpolation

CHAPTER 3 REFERENCES

[Baucher and Hassab, 1981]

R.E. Baucher and J.C. Hassab, "Analysis of Discrete Implementation of Generalized Cross-Correlator", IEEE Trans. Acous. Sp. Sig. Proc., Vol. ASSP-29, June 1981, pp. 609-611.

[Duda and Hart, 1973]

R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973.

[IMSL, Inc., 1982]

International Mathematics and Statistics Library User Manual, Version 9, IMSL, Inc., 7500 Bellaire Blvd., Houston, TX, 77036.

[Lucas and Kanade, 1981]

B.D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", **Proc. of the Workshop on Image Understanding, IEEE Press**, 1981.

[Kumar et al, 1982]

B.V.K. Vijaya Kumar, D. Casasent and A. Goutzoulis, "Fine Delay Estimation with Time-Integrating Correlators", **Applied Optics, Vol. 21**, November 1982, pp. 3855-3863.

4. Image Understanding Techniques for 3-D Scene Interpretation

4.1 Introduction

In this chapter, we present results in two aspects of the 3-D change detection task: the low-level problem of analyzing images, and the high-level problem of representing, constructing, and updating the 3-D scene model. The input to the system is a sequence of high resolution aerial images of an urban scene, such as might be obtained from space-based sensors. We have developed techniques that will be useful in monitoring changes and developments at cultural sites, such as urban areas and military bases.

For the low-level processing, we describe techniques for extracting building structures from the images. First, linear segments and junctions are extracted. These segments and junctions are then assigned to an a priori structural model of buildings. A search is then performed to look for new line segnments predicted by the model.

This technique makes heavy use of junctions in the image, which are assumed to correspond to building corners. The procedure for forming junctions from line segments in the image can be made efficient by dividing the image into a number of small areas called sectors, and storing with each sector a list of lines in its area. To search for lines within a window that might form a junction, only the line lists of sectors containing the window need be searched. We describe experiments which determine how to divide the image so as to minimize the search time.

For the high-level processing, we describe techniques for representing, constructing, and updating the 3D scene model. The scene model is a surface-based description of an urban scene, and is incrementally acquired from a sequence of images. Each view of the scene undergoes analysis which results in a 3D wire-frame description that represents portions of edges and vertices of buildings. We describe how these wire frames are used to construct and modify the model.

4.2 Extracting Lines and Junctions from Images

This section briefly describes our techniques for extracting linear segments and junctions from complex aerial images of urban scenes.

To extract linear features, a 3x3 Sobel operator is first used to extract edge points, as shown in Fig. 4-1. Then the edges are thinned using a modified Nevatia and Babu algorithm [Nevatia and Babu 80], as shown in Fig. 4-2. The resulting edge points are linked and approximated by piecewise linear segments. The method used to fit linear segments to a set of linked points is based on iterative end-point fitting [Duda and Hart 73]. However, since this method determines a line using only two end points, the line equation for the set of points is recalculated using least squares. Finally, short lines are discarded. The resulting line image is shown in Fig. 4-3.

In the next step, we extract junctions from the line image. A junction is a group of line segments (*legs*) in the image that meet at a point, and often arises from a vertex in the scene. We consider the following four junction types: L, ARROW, FORK, and T. To find junctions, a 5x5 window around each end point of each line is searched for ends of other lines. Lines in the window that are close and nearly parallel are combined into a single line. Then, if the window contains the ends of three lines, the lines are classified as an ARROW, FORK, or T junction depending on the angles between the lines. The position of the junction point is the middle of the three end points. If the window contains the ends of two lines, they are classified as an L junction: the intersection of the two lines determines the position of the junction. If the window contains more than three lines, each set of two lines is assumed to form a distinct L junction. Junctions that have been found in this manner are labeled in Fig. 4-4. Notice that many of the junctions correspond to building corners.

4.3 A Model Based Search to Find Building Boundaries in Aerial Images

4.3.1 Introduction

A system (Fig. 4-5) has been created to extract portions of building boundaries from aerial images. Line features are first extracted from the image. Junctions are then formed with the line features and the line features are assigned to structural models of a building based on the junctions. A search is made for missing line features in each structural model. This project is motivated by an observation of human interpretation of the extracted lines from an aerial image. A human interpreter tends to pick a collection of lines which seem to correspond to part of a building and then searches for less obvious lines which would support the hypothesized building. For example, a human may initially pick the three lines corresponding to a corner of a building and then search for neighboring corners and their lines. Or a human may pick four lines forming a parallelogram to be the roof of a building and then search for the missing lines to form a parallelepiped.

A similar strategy is used in the system. In perspective projection all line features corresponding to vertical edges of buildings will converge on a single point (called the vertical vanishing point) and can be identified in this manner. The system is based on a simple model of buildings which has a strong dependence on vertical edges. The word vertical will refer to vertical in space, a known direction in the image. The model is a serial list of rectangular shaped faces. A face may contain up to three lines: a vertical edge and top and bottom

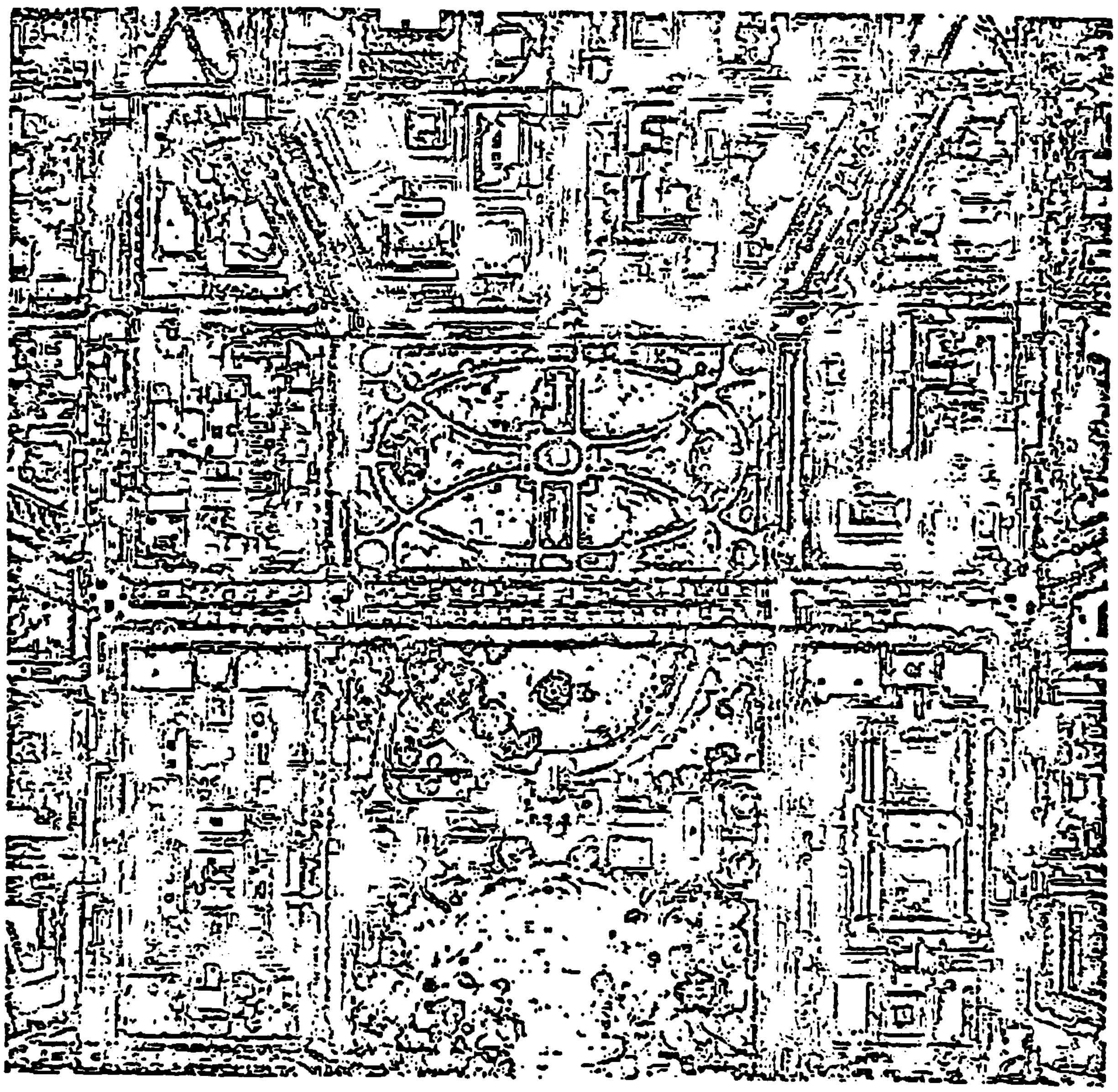


Figure 4-1: Edges resulting from a Sobel operator applied to an aerial image of a region of Washington, D.C.

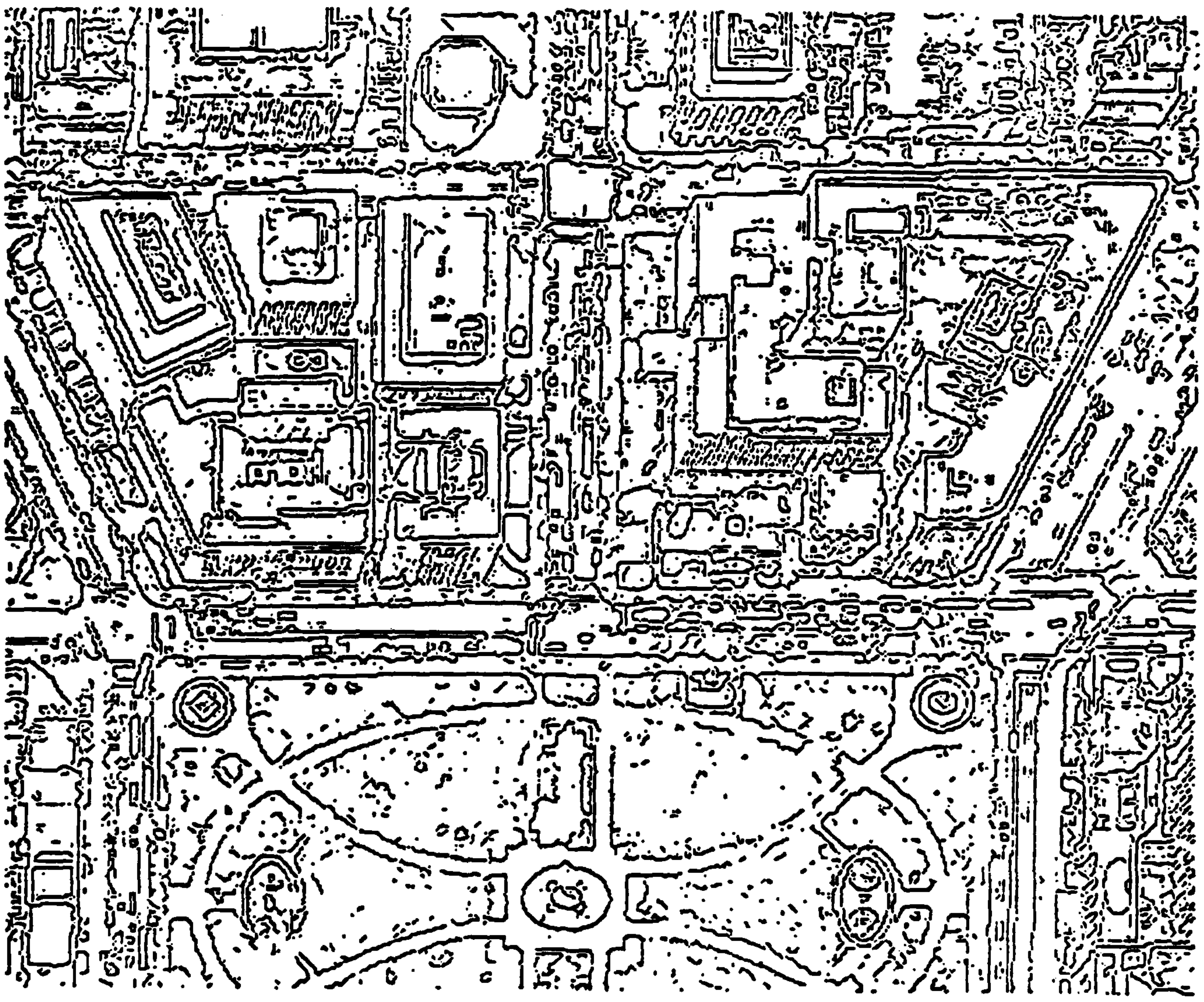


Figure 4-2: Result of thinning the edges in Fig. 4-1. Results for the upper middle part of the image in Fig. 4-1 are shown here.

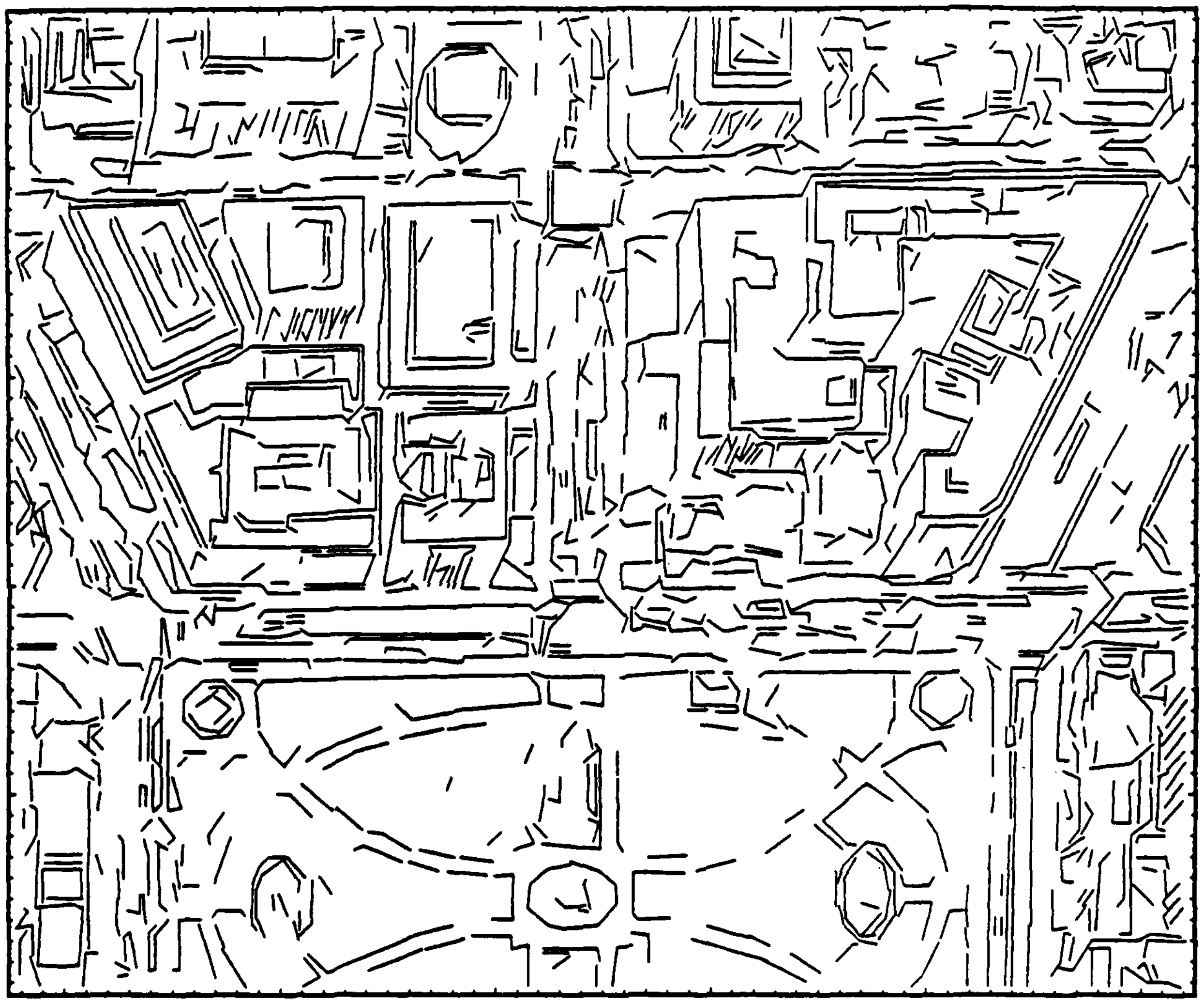


Figure 4-3: Linear segments fitted to the edge points of Fig. 4-2 after they are linked.

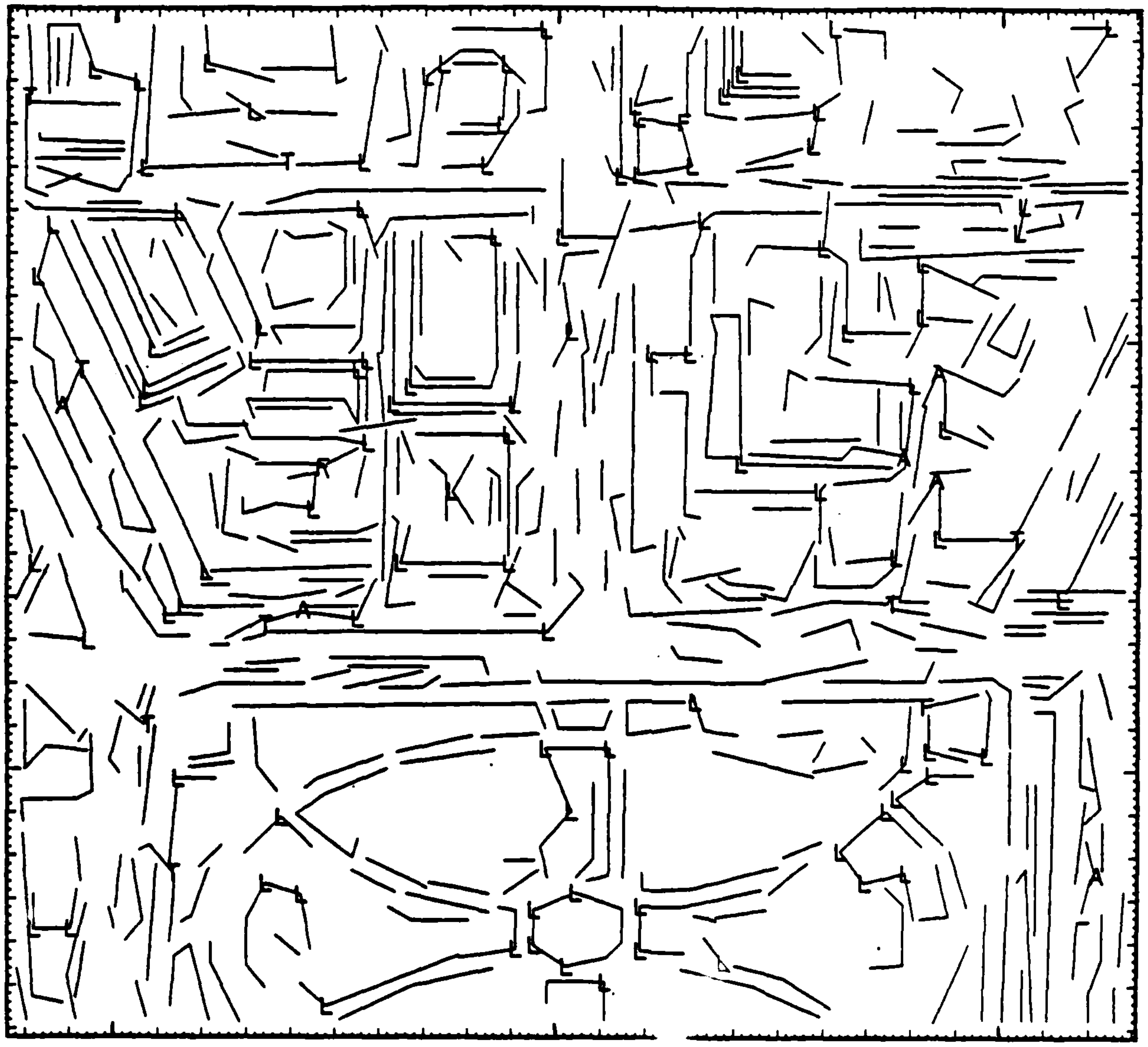


Figure 4-4: Result of classifying junctions in a different version of the line images than shown in Fig. 4-3. Junctions are classified as L, A (arrow), F (fork), or T.

edges. Line features are initially assigned to the model based on the junctions connecting them. Junction types L, A, F and T are used. Since one line in an A or F junction is constrained to be a vertical edge, a unique location in the model for all three lines is determined. A collection of lines can be interconnected by a number of junctions. Once the three lines from a single A or F junction have been added to the model, the remaining lines can be added to the model using the same vertical edge constraint. Section 4.3.2 contains the model definition and a more detailed description of how collections of lines are assigned to it.

The collection of lines assigned to an instance of the model will be refered to as a structure. Once the lines linked by junctions have been formed into structures a search begins for other lines belonging to the same structure. Most of the locations to search are determined by missing lines in the model. A few other probable locations are also checked. The search is first done for lines which do not have junctions and were not linked into structures. The second time, a Hough transform is used to extract new line features. Most of the searching strategy is described in section 4.3.2. Section 4.3.3 is mainly a description of the Hough transform.

Section 4.3.4 presents results from several images of the Washington, DC area. Section 4.3.5 summarizes some conclusions drawn from the project.

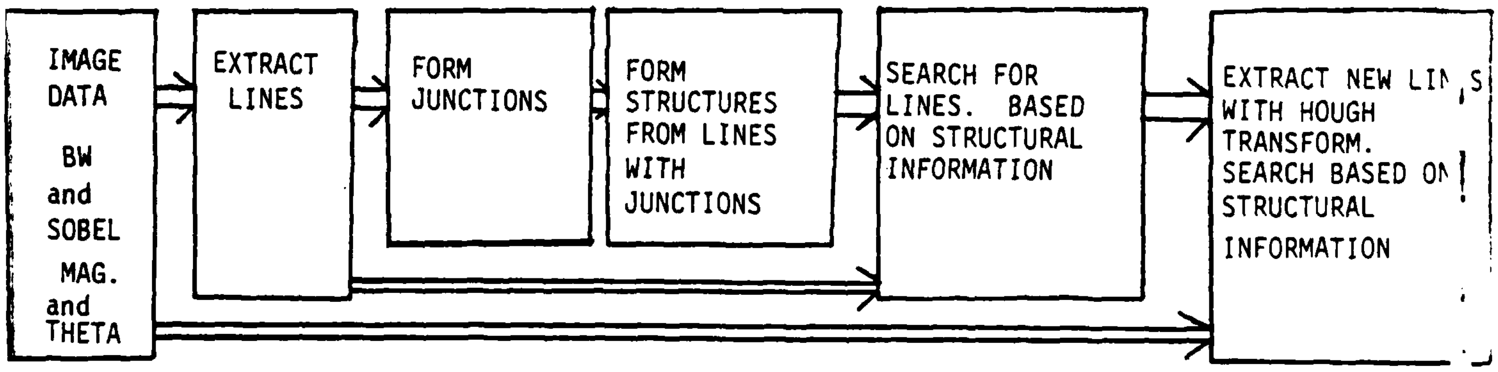


IMAGE DATA
BW and SOBEL MAG. and THETA
EXTRACT LINES
FORM JUNCTIONS
FORM STRUCTURES FROM LINES WITH JUNCTIONS
SEARCH FOR LINES. BASED ON STRUCTURAL INFORMATION
EXTRACT NEW LINES WITH HOUGH TRANSFORM. SEARCH BASED ON STRUCTURAL INFORMATION

Figure 4-5: System Block Diagram

4.3.2 Forming Structures

This section describes the model and how a collection of lines is assigned to it. There are two ways a structure can be formed. In the first, a structure is created in two steps. The first step creates structures based on the junctions linking a set of lines. The second step adds more lines via a search based on the structural information. The second method of forming a structure is from vertical edges extending down from the three points of a single L junction. This was an attempt to improve performance and should be replaced by a heuristic junction finder.

4.3.2.1 The Building Model

The model of a building is based on three assumptions:

- Only A, F, and L junctions are permitted.
- The vertical vanishing point is known. All lines converging toward this point are assumed to be vertical.
- Buildings look like polyhedral layer cakes.

An instance of the model is shown in Fig. 4-6. The model consists of a serial list (possible circular) of faces. A face corresponds to the side of a building. A face is the shape of a parallelogram and contains up to three lines. One line is the top edge, another the bottom edge, and the third is the (right most) vertical edge of that side of the building. A face may be visible or not visible (in which case only the top line is visible). The top surface (roof) of a structure and the bottom surface (at the ground plane) are indirectly described in the model by the set of top and bottom lines in the faces. Some complex buildings could therefore be represented by several instances of this model stacked vertically.

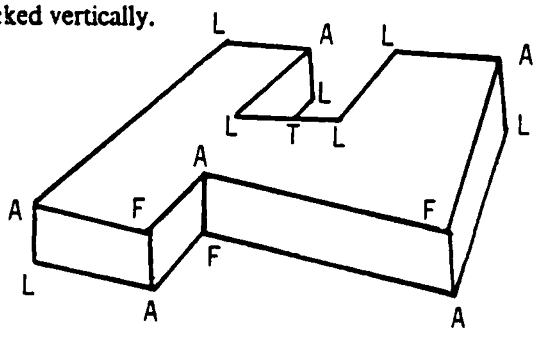


Figure 4-6: General instance of the model

4.3.2.2 Adding Lines to the Model According to Junctions

There are two ways a collection of lines connected by junctions are formed into structures. The first way is the simple case of a series of L junctions forming a closed polygon. All the lines belong to the tops of faces and the faces form a closed loop. The second way to form a structure considers all the remaining cases and is described below.

A result of the model definition is that A and F junctions have a unique position in the model. One of the three lines in an A or F junction is required to be a vertical edge. This constrains the two remaining lines to belong to the top or bottom of the two faces. By considering the orientation of these two lines with the vertical one, their spatial relation is determined. If none of the three lines in the junction is vertical, the three lines are not assigned to a structure. All the possibilities for a single A or F junction are listed below.

- The shaft of an A junction is vertical. The barbs belong to either the top or bottom of their faces.
- A barb of an A junction is vertical. The other two lines must be the tops of their faces. One face is not visible.

- Any F junction. The two non-vertical lines both belong to either the top or bottom of their faces.

When a new A or F junction is located, its three lines form a new structure. Other lines may be connected to the new structure by other junctions. To find these other lines, we check if a junction exists at the end of each line in the A or F junction. If a new junction exists at that end, an attempt is made to add its lines to the structure. If the new junction is of type A or F there is only one possible location for it in the model. If the new junction is an L, the possible location of the single new line in the structure is easily determined by the angle of the new line with respect to the (possibly hypothetical) vertical edge at that corner. T junctions are caused by occlusions and do not need to be considered if the two lines in the crossbar of the T junction are merged into one line. The T junction's lines are assigned to locations in the structure based on the type of junction at the other end of the lines. Lines from a new junction are not added if their location in the structure conflict with lines currently contained in the structure. If a line is encountered which belongs to another structure, it is not added. A search is not made to determine the best fit structure for a given collection of lines. Two simple rules are used instead.

- Form structures from F junctions first, then do it again with the A junctions. This is because F junctions tend to be more prominent than A junctions.

- Follow lines closer to the top of a structure first. The most prominent lines usually correspond to the roof top of a building.

4.3.2.3 Adding Lines to a Structure Based on the Structure

The current structures contain lines linked by junctions. In this step the information in the structure is used to search for more lines. These new lines are not required to have junctions. There are a few cases to consider when searching for new lines.

- Search for any missing line in a face.

- If the faces do not form a closed loop, s ch for the top line of a new face at the two ends of the list of faces. This line must be parallel to the second face from that end.

- If the faces do not form a closed loop, try to find a line to connect the top.

In each of these cases the predicted location, length and angle of the missing line is determined by the current lines in the structure. The search is done in a rectangular window. The size of the window is determined by the length and angle of another line in the structure which depends on the location of the missing line. Any new line produced by the search must be within a tolerance of the other line, near a corner of the structure, and of a minimum length. If the new line has junctions at either end, an attempt is made to add the lines connected to these junctions to the structure. A special case occurs if a windowed line already belongs to another structure. The structure which results from adding the windowed line (and all lines connected to it) must be larger than the structure originally containing the windowed line. If several lines are windowed, we choose either the one which forms the largest structure or simply the longest line. When a structure is destroyed because some of its lines merged with a larger structure, an attempt is made to reform structures from the remaining lines.

4.3.2.4 Searching for Vertical Edges Around L Junctions

The method of creating structures which has just been described depends mostly on the existance of junctions with 3 legs containing one vertical leg. In an attempt to improve this situation a search for vertical legs extending from all L junctions is made. Three locations are searched: the junction point and the end of the two legs. If a vertical line is found, the three lines form a structure. If the vertical line is at the end of a leg, the structure is ambiguous since two possibilities exist. Since it cannot be determined if one of the faces is visible or not, the system simply chooses the visible case. This search was implemented because it is quick and simple. Similar results should be obtained with a heuristic junction finder.

4.3.3 Extracting New Lines

4.3.3.1 Searching

The next task is to extract new lines from the raw image data based on information in the structures. This step is similar to what was previously described, except that the search is in the pixel domain rather than the domain of extracted lines. The search locations are identical. The search is performed via a Hough transform which returns a list of line segments found in the search window.

4.3.3.2 The Hough Transform

The Hough transform is a method to extract lines by mapping edge points from point space to the parameter space of lines. The Hough transform uses the equation for a line:

$$r = x\cos(\theta) + y\sin(\theta)$$

This equation maps a single point in (x,y) space to a sinusoid in (r,θ) space. The sinusoid represents all the lines passing through the point (x,y). Separate sinusoids are plotted for each edge point at (x,y) in a windowed area. The (r,θ) location containing the largest number of sinusoids represents the best fit line to the set of points in the windowed area.

The Hough transform implemented in the system uses data from the sobel edge operator. The sobel edge operator (Fig. 4-7) produces values for edge magnitude and direction.

The (r,θ) space is quantized as an accumulator containing a number of cells. A cell measures 1 pixel by 5 degrees. A cell is incremented by the sobel magnitude of the pixel point mapping the sinusoid. Only cells within 30 degrees of the sobel angle of the pixel point are incremented. The value of r can be positive or

$$\Delta_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \Delta_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$$\text{magnitude} = \sqrt{\Delta_y^2 + \Delta_x^2}$$

$$\text{angle} = \operatorname{atan2}(\Delta_y, \Delta_x)$$

Figure 4-7: Sobel Edge Operator

negative. Both the sobel angle and the accumulator cells range from 0 to 2π. This has the effect of separately considering lines with intensity variations across them in either direction. The extracted lines are obtained by choosing all cells exceeding 90 percent of the maximum cell. To prevent extracting lines in a window containing only noise, the value of the maximum cell must be above an absolute minimum value . The final step is to determine line segments for the line represented by each cell. The pixel points contributing to each cell are linked into one or more segments and a line is fitted to each segment.

4.3.4 Results

The results from several images of the Washington, DC area are shown in Figs. 4-8 to 4-12. Figs. 4-8, 4-9, and 4-10 are three views of the same scene. Fig. 4-12 is from a magnified version (obtained using a cubic interpolation algorithm) of the image used to generate Fig. 4-11. Each of the figures consists of three parts. Part (a) shows all the lines originally extracted from the image. The cross hatching lines are spaced 20 pixels apart. Junctions of type L, A, F and T are labeled. Part (b) shows only the lines assigned to structures. The heavy lines are produced by the Hough transform. The lines are numbered. Part (c) refers to the line numbers to describe the organization of each structure. The structures are numbered. Three things are listed for each structure in part (c): the number of lines contained in the structure, whether the faces form a closed loop (labeled as 'top is connected'), and a list of the faces. Four items are listed for each face: the top, bottom, and (rightmost vertical) side line numbers, and whether the face is visible (labeled as 'is exposed') or not visible. Faces are listed in counter-clockwise order when a structure is viewed from above.

The overall performance of the system is fairly good. For example, fourteen structures are contained in Fig. 4-8. Seven of them (structures 1, 2, 4, 5, 6, 7, 12) correspond correctly with buildings. Two (structures 9, 14) correspond only partially, and five (structures 3, 8, 10, 11, 13) do not correspond to any building. Some observations on the results are listed here.

- Although many structures correspond to the desired building edges in the scene, a few structures do not correspond at all. For example, the area occupied by structure 3 in Fig. 4-8 does not contain a building. Structure 8 is a result of the sidewalk and street corner.

- Some of the structures result from the ambiguous case of the vertical edge search at the ends of an L junction. An example is structure 11 in Fig. 4-12. (The incorrect vertical lines are a result of the Hough transform implementation. It does not have a minimum threshold and extracts lines from the noise in the image areas.)

- Structures often overlap. In one case two structures may each describe a different two thirds of the same building. *Structures 2 and 11 in Fig. 4-9 and structures 2 and 3 in Fig. 4-11 are* examples.

- A large number of structures (often containing a substantial number of lines) result from searching for vertical edges around L junctions. For example, structures 6, 7 and 12 in Fig. 4-8.

- Multiple lines are extracted for a single edge in the scene. This is a result of two deficiencies in the system. The first is caused by short lines. In this case a search window may overlap and extract a scene edge already contained in the structure. For example, a face is missing a bottom line, but its vertical line is very short. The search window for the bottom line may include the location of the top line. The extracted bottom line may actually result from the top of the building. The second deficiency arises when the faces do not form a closed loop. The two end faces are not aware of each other. For example, four lines almost form a rectangle (the roof of a building) but two lines are not close enough to be linked as a single corner. When searching for new top lines connected to the end faces, the system does not realize the extracted line corresponds to the top line of the face at the opposite end.

4.3.5 Conclusions

The results have shown it is possible to extract substantial portions of some buildings. The model works well with isolated simple buildings and seems sufficient for the simple urban environment which has been studied. Several improvements could be made.

- A larger number of junctions with 3 legs is highly desirable since the model depends on them so much. The junction finder can be improved to produce more junctions in two ways. The first is to use the constraint that one of the legs must be a vertical edge. Evidence of this is the large number of structures originating from the search for vertical edges at the ends of L junctions. The second improvement is to use the grayscale intensity along each side of a line. The intensity on one side of all the lines forming a single face should be similar. The junction finder used in the system only forms junctions with 3 legs if no more than three line endpoints lie within a certain area. If a vertical edge heuristic and intensity information are used, a selection could be made from a group of four or more lines in an area to form one or more junctions with 3 legs.

- The extracted lines usually do not correspond exactly to the edge in the image. They tend to be shorter. A very useful idea is to try extending all lines at each end that does not have a junction.

- The system does not perform an extensive or complex search to find the best fit structure for a set of lines. A more complex search should find a few additional structures. Using intensity information to select lines may provide a more accurate choice when many lines are found in an area.

- An interesting extension of this work is to use structures from multiple views of a scene. Matching would have to be performed. The global set of structures for the scene could be used to search specific images for missing or hypothesized line features or even entire buildings. Spatial relations could be used to find holes between nearby buildings which may contain a missing building. Especially in the urban scenes studied, knowledge of the location of roads would be very helpful.

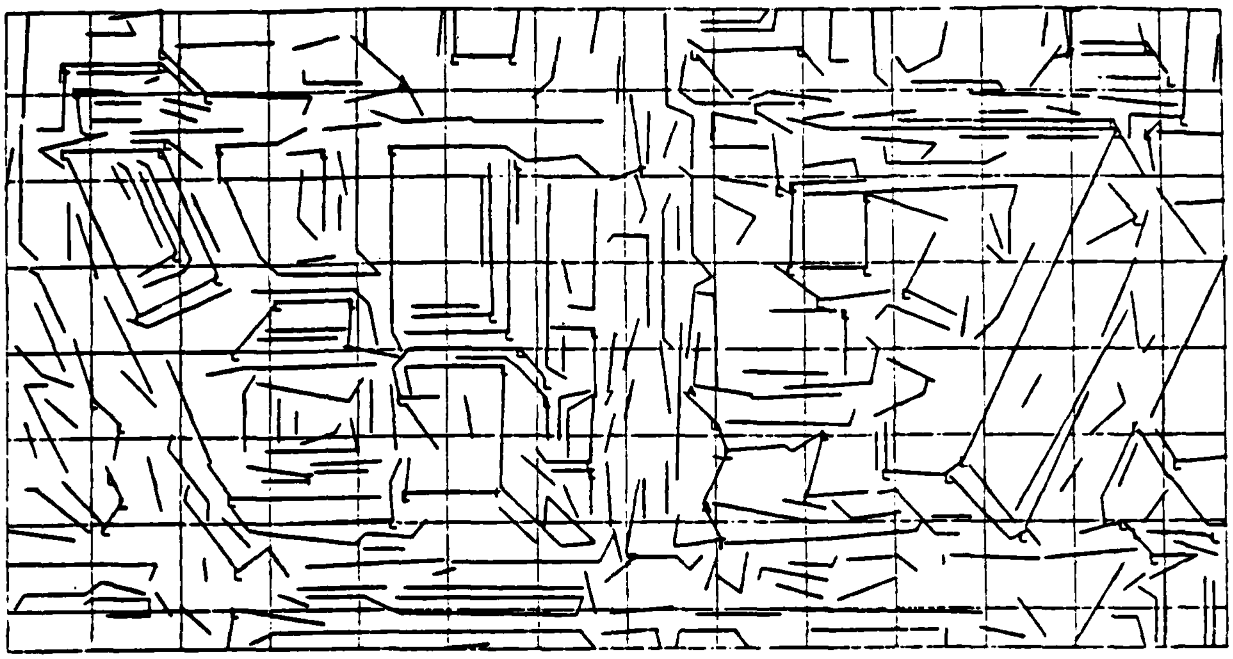


Figure 4-8a: Front Right View

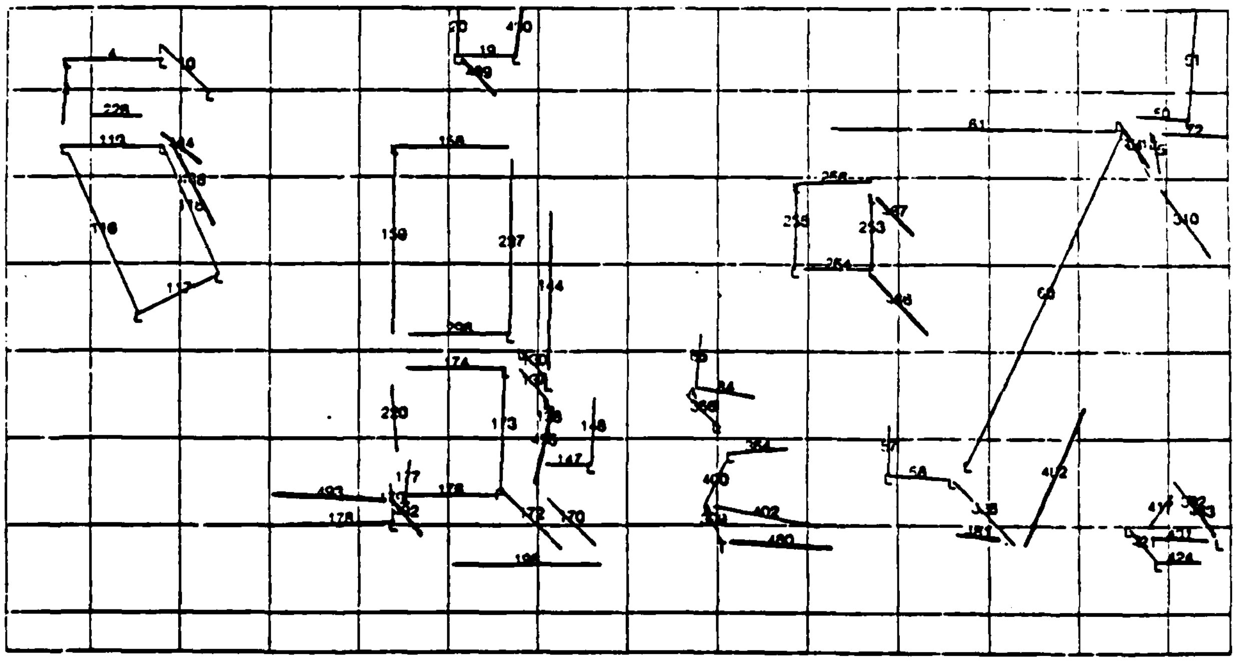


Figure 4-8b: Front Right View

```
STRUCTURES ARE:
---------- ----

Structure has   6 lines:
          Top IS connected.
          top=253 bot=... next side=487   Face IS exposed
          top=256 bot=... next side=...   Face is not exposed
          top=255 bot=... next side=...   Face is not exposed
          top=254 bot=... next side=486   Face IS exposed

Structure has   3 lines:
          Top is not connected.
          top= 86 bot=... next side=366   Face is not exposed
          top= 84 bot=... next side=...   Face IS exposed

Structure has   5 lines:
          Top is not connected.
          top=364 bot=... next side=...   Face is not exposed
          top=400 bot=... next side=399   Face is not exposed
          top=402 bot=480 next side=...   Face IS exposed

Structure has   5 lines:
          Top IS connected.
          top=118 bot=... next side=488   Face IS exposed
          top=119 bot=... next side=...   Face is not exposed
          top=116 bot=... next side=...   Face is not exposed
          top=117 bot=... next side=...   Face IS exposed

Structure has   9 lines:
          Top IS connected.
          top=177 bot=... next side=...   Face is not exposed
          top=176 bot=198 next side=172   Face IS exposed
          top=173 bot=138 next side=137   Face IS exposed
          top=174 bot=... next side=...   Face is not exposed
          top=220 bot=... next side=...   Face is not exposed

Structure has   8 lines:
          Top is not connected.
          top= 57 bot=... next side=...   Face is not exposed
          top= 58 bot=481 next side=      Face IS exposed
          top= 60 bot=482 next side=      Face IS exposed
          top= 61 bot=... next side=...   Face is not exposed

Structure has   6 lines:
          Top IS connected.
          top=297 bot=144 next side=...   Face IS exposed
          top=158 bot=... next side=...   Face is not exposed
          top=159 bot=... next side=...   Face is not exposed
          top=298 bot=... next side=130   Face IS exposed

Structure has   6 lines:
          Top IS connected.
          top=483 bot=... next side=392   Face is not exposed
          top=417 bot=... next side=421   Face is not exposed
          top=431 bot=424 next side=...   Face IS exposed

Structure has   5 lines:
          Top is not connected.
          top=... bot=... next side=484   Face is not exposed
          top=226 bot=... next side=...   Face is not exposed
          top=  2 bot=... next side=...   Face IS exposed
          top=  4 bot=... next side= 10   Face IS exposed

Structure has   3 lines:
          Top is not connected.

          top=... bot=... next side=310   Face is not exposed
          top= 66 bot=... next side=...   Face IS exposed
          top= 72 bot=... next side=...   Face IS exposed

Structure has   4 lines:
          Top is not connected.
          top=486 bot=... next side=170   Face is not exposed
          top=147 bot=... next side=...   Face IS exposed
          top=148 bot=... next side=...   Face IS exposed

Structure has   4 lines:
          Top is not connected.
          top= 20 bot=... next side=489   Face is not exposed
          top= 19 bot=... next side=...   Face IS exposed
          top=490 bot=... next side=...   Face IS exposed

Structure has   3 lines:
          Top is not connected.
          top=... bot=... next side=491   Face is not exposed
          top= 50 bot=... next side=...   Face IS exposed
          top= 51 bot=... next side=...   Face IS exposed

Structure has   4 lines:
          Top is not connected.
          top=178 bot=... next side=...   Face IS exposed
          top=179 bot=... next side=492   Face IS exposed
          top=493 bot=... next side=...   Face is not exposed
```

Figure 4-8c: Front Right View

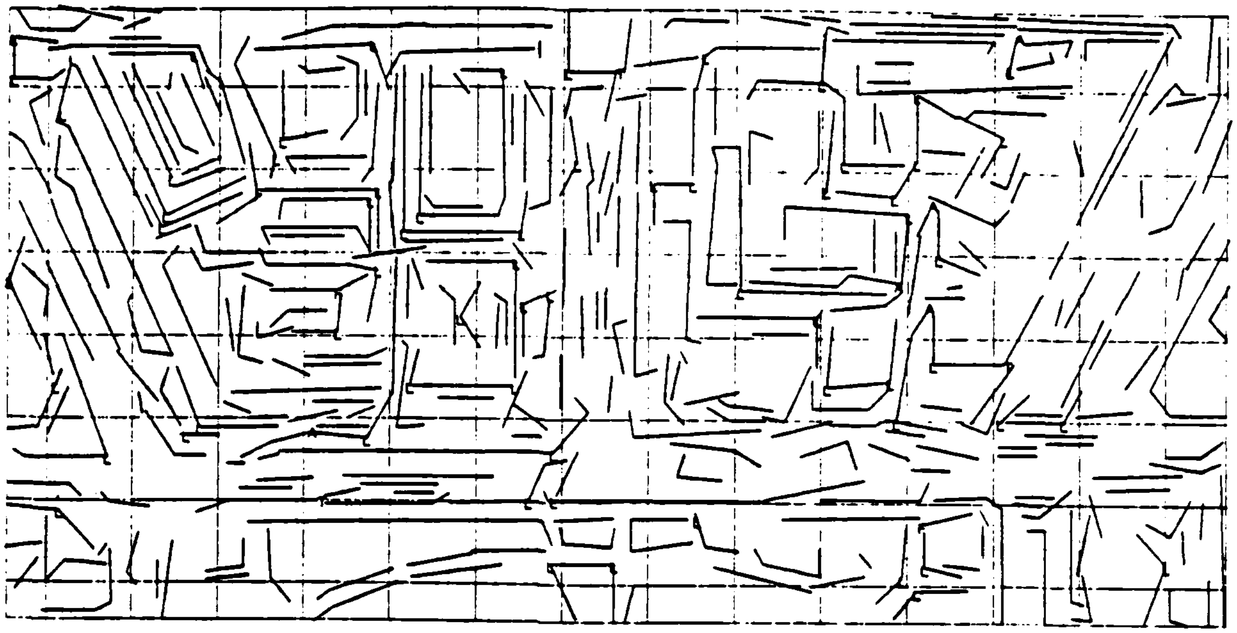


Figure 4-9a: Front Left View

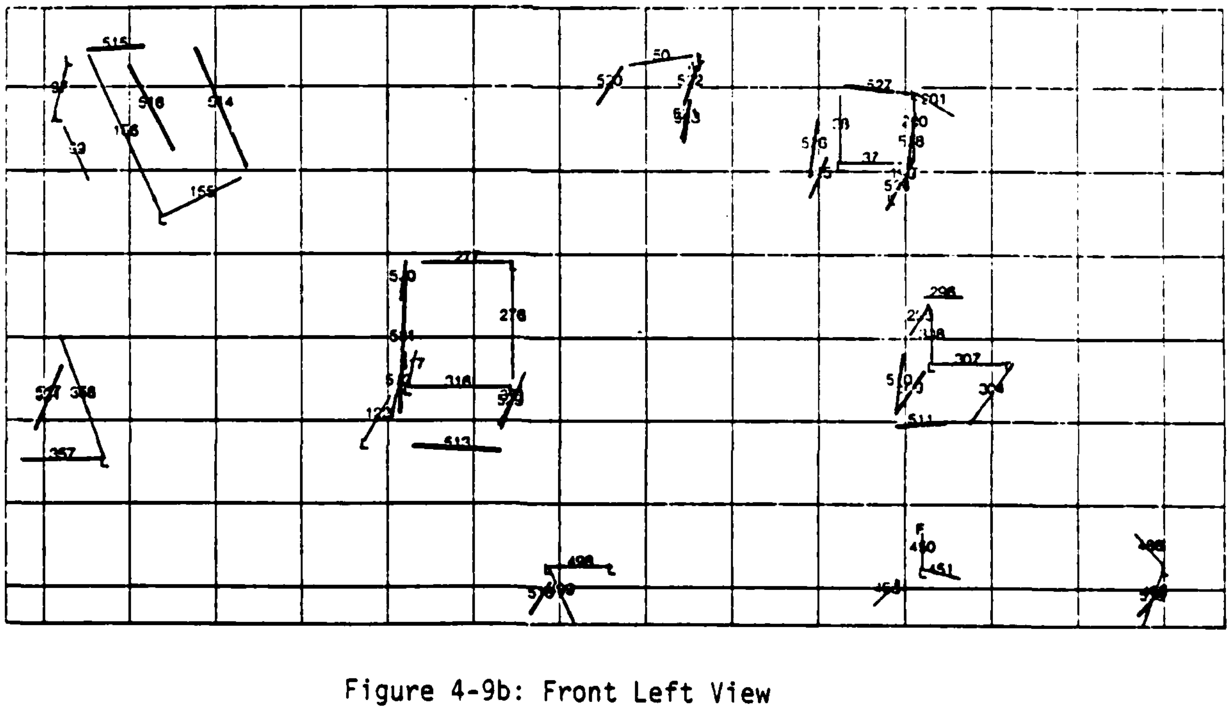


Figure 4-9b: Front Left View

```
STRUCTURES ARE:
---------- ----

Structure has   8 lines:
          Top is not connected.
          top=296 bot=... next side=295   Face is not exposed
          top=306 bot=510 next side=509   Face IS exposed
          top=307 bot=511 next side=304   Face IS exposed


Structure has   8 lines:
          Top is not connected.
          top=... bot=... next side=512   Face is not exposed
          top=317 bot=... next side=122   Face IS exposed
          top=316 bot=513 next side=371   Face IS exposed


Structure has   3 lines:
          Top is not connected.
          top=460 bot=... next side=463   Face IS exposed
          top=451 bot=... next side=...   Face IS exposed


Structure has   3 lines:
          Top is not connected.
          top=201 bot=... next side=...   Face is not exposed
          top=200 bot=... next side=199   Face IS exposed


Structure has   7 lines:
          Top is not connected.
          top=516 bot=... next side=...   Face is not exposed
          top=515 bot=... next side= 97   Face is not exposed
          top=156 bot= 99 next side=...   Face IS exposed
          top=155 bot=... next side=...   Face IS exposed
          top=514 bot=... next side=...   Face is not exposed


Structure has   4 lines:
          Top is not connected.
          top=517 bot=... next side=361   Face is not exposed
          top=356 bot=... next side=...   Face IS exposed
          top=357 bot=... next side=...   Face is not exposed
```

```
Structure has   3 lines:
          Top is not connected.
          top=498 bot=... next side=618   Face is not exposed
          top=499 bot=... next side=...   Face IS exposed


Structure has   3 lines:
          Top is not connected.
          top=486 bot=... next side=519   Face IS exposed
          top=485 bot=... next side=...   Face IS exposed


Structure has   6 lines:
          Top is not connected.
          top=... bot=... next side=620   Face is not exposed
          top= 50 bot=... next side=522   Face IS exposed
          top= 47 bot=523 next side=621   Face IS exposed


Structure has   7 lines:
          Top IS connected.
          top=528 bot=... next side=...   Face is not exposed
          top=527 bot=... next side=...   Face is not exposed
          top= 38 bot=526 next side=526   Face IS exposed
          top= 37 bot=... next side=624   Face IS exposed


Structure has   5 lines:
          Top is not connected
          top=531 bot=... next side=530   Face is not exposed
          top=277 bot=... next side=...   Face IS exposed
          top=278 bot=... next side=529   Face IS exposed
```

Figure 4-9c: Front Left View

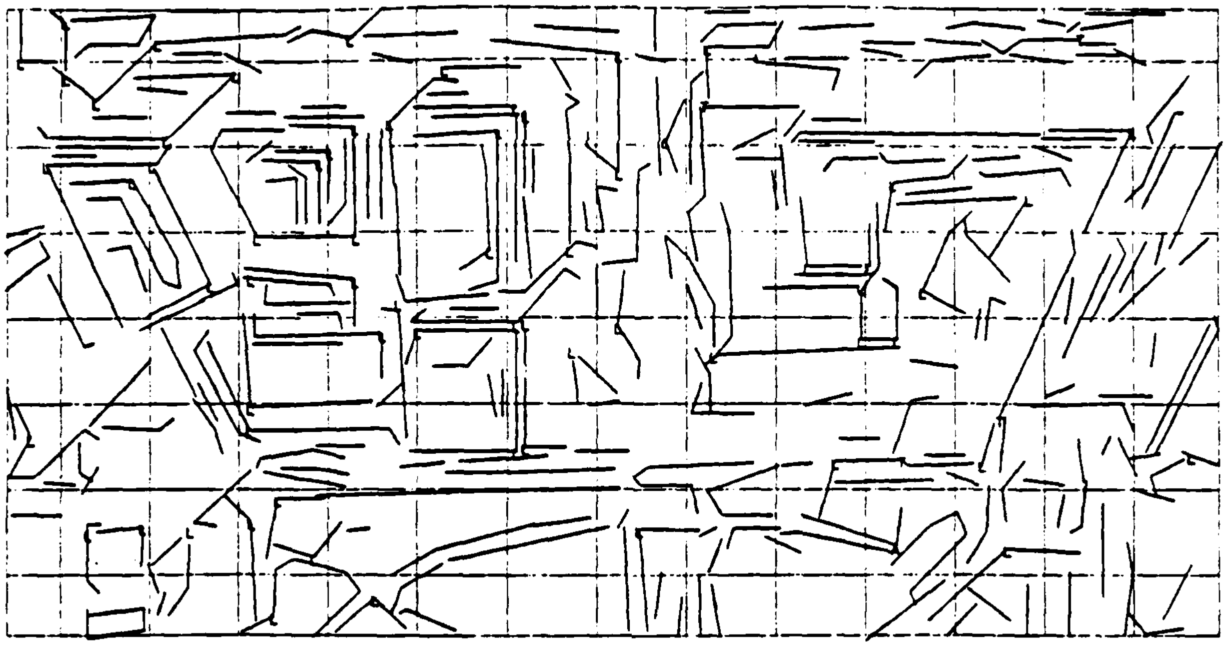


Figure 4-10a: Rear Right View

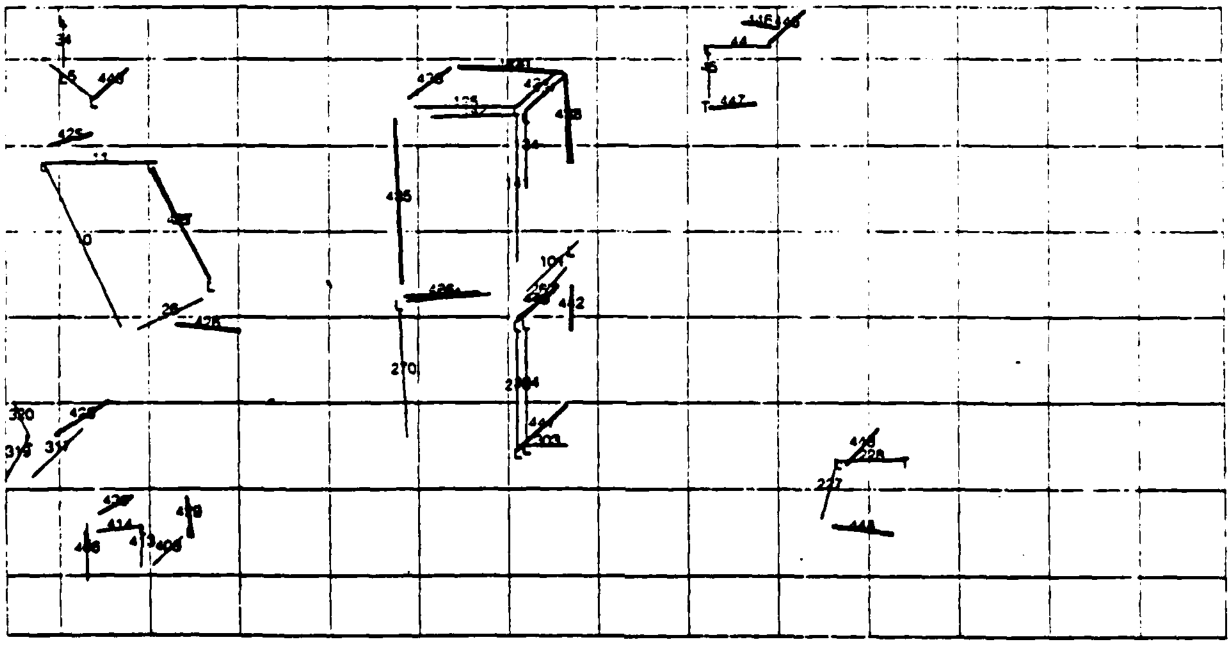


Figure 4-10b: Rear Right View

```
STRUCTURES ARE:
---------- ----

Structure has   7 lines:
         Top is not connected.
         top=428 bot=... next side= 28   Face IS exposed
         top= 10 bot= 25 next side=425   Face IS exposed
         top= 11 bot=... next side=...   Face is not exposed
         top=427 bot=... next side=...   Face is not exposed

Structure has   3 lines:
         Top is not connected.
         top=... bot=... next side=101   Face is not exposed
         top=124 bot=... next side=...   Face IS exposed
         top=270 bot=... next side=...   Face is not exposed

Structure has   4 lines:
         Top is not connected.
         top=... bot=... next side=317   Face is not exposed
         top=319 bot=... next side=428   Face IS exposed
         top=320 bot=... next side=...   Face IS exposed

Structure has   6 lines:
         Top is not connected.
         top=... bot=... next side=408   Face is not exposed
         top=413 bot=429 next side=...   Face IS exposed
         top=414 bot=... next side=430   Face IS exposed
         top=406 bot=... next side=...   Face is not exposed

Structure has   8 lines:
         Top is not connected.
         top=141 bot=432 next side=431   Face IS exposed
         top=125 bot=434 next side=433   Face IS exposed
         top=436 bot=... next side=...   Face is not exposed
         top=436 bot=... next side=...   Face is not exposed

Structure has   5 lines:
         Top is not connected.
         top=134 bot=438 next side=437   Face IS exposed
         top=142 bot=439 next side=...   Face IS exposed

Structure has   5 lines:
         Top is not connected.
         top=... bot=... next side=441   Face is not exposed
         top=273 bot=442 next side=440   Face IS exposed
         top=267 bot=... next side=...   Face IS exposed

Structure has   4 lines:
         Top is not connected.
         top=228 bot=... next side=443   Face IS exposed
         top=227 bot=... next side=...   Face is not exposed
         top=444 bot=... next side=...   Face is not exposed

Structure has   5 lines:
         Top is not connected.
         top=... bot=... next side=445   Face is not exposed
         top= 44 bot=446 next side=...   Face IS exposed
         top= 46 bot=... next side=...   Face is not exposed
         top=447 bot=... next side=...   Face is not exposed

Structure has   3 lines:
         Top is not connected.
         top=... bot=... next side=448   Face is not exposed
         top=  5 bot=... next side=...   Face IS exposed

Structure has   3 lines:
         Top is not connected.
         top=303 bot=... next side=...   Face IS exposed
         top=304 bot=... next side=449   Face IS exposed
```

Figure 4-10c: Rear Right View

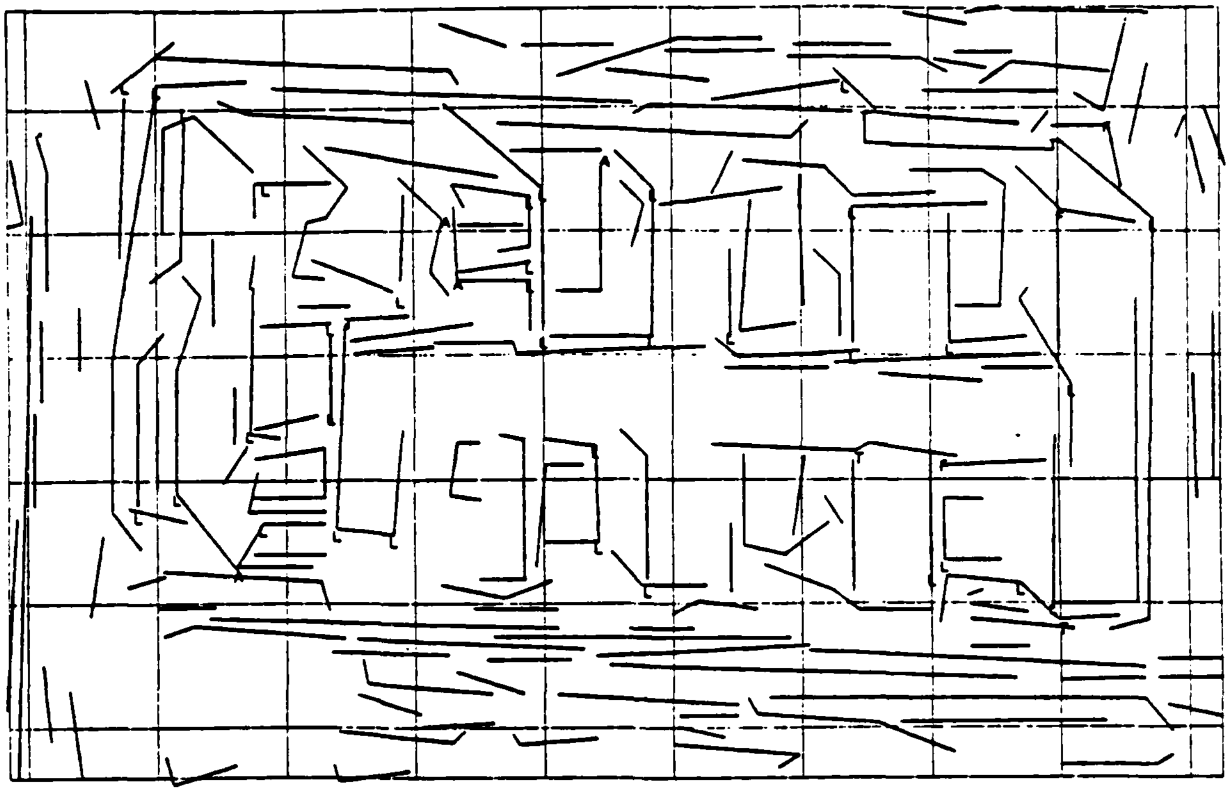


Figure 4-11a: Complex Building

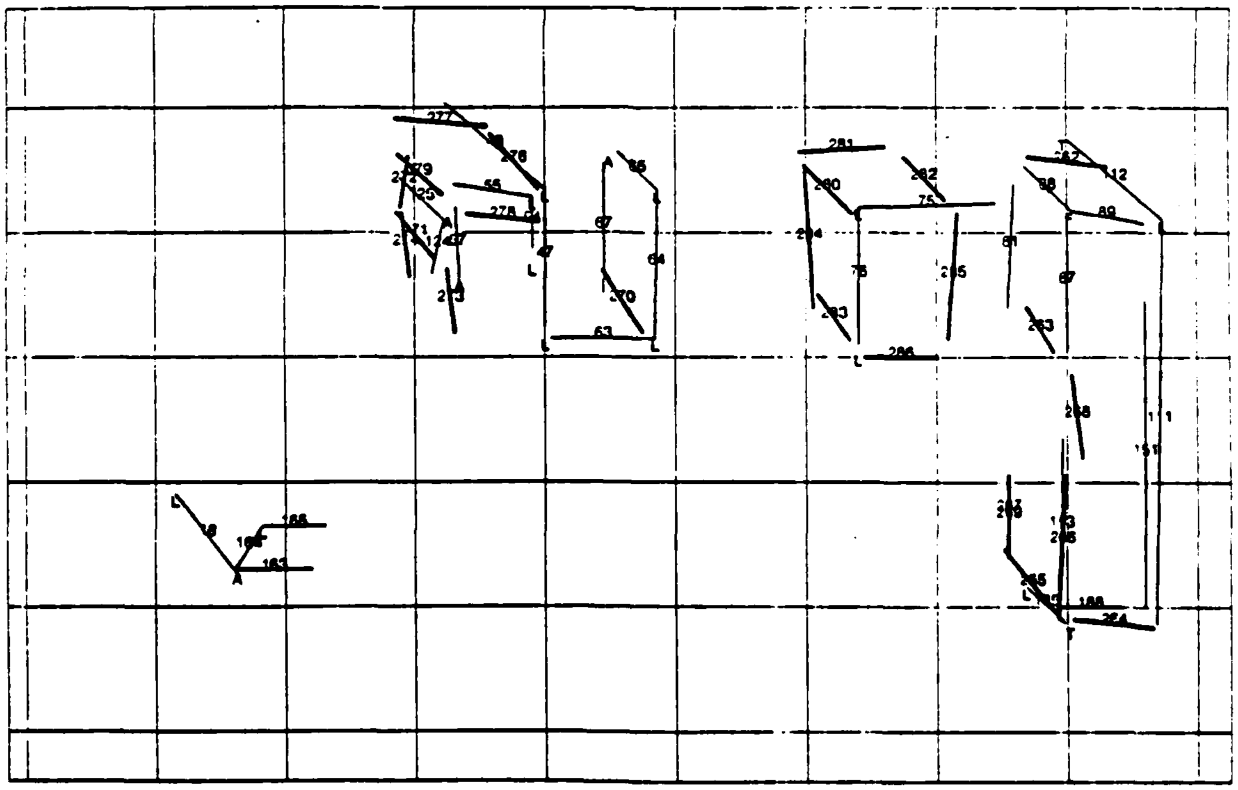


Figure 4-11b: Complex Building

```
STRUCTURES ARE:
---------- ----

Structure has   4 lines:
          Top is not connected.
          top=166 bot=... next side=...    Face IS exposed
          top=164 bot=... next side= 18    Face IS exposed
          top=163 bot=... next side=...    Face is not exposed


Structure has  13 lines:
          Top IS connected.
          top=266 bot=267 next side=265    Face IS exposed
          top=264 bot=... next side=...    Face is not exposed
          top=111 bot=... next side=112    Face is not exposed
          top= 89 bot=262 next side= 88    Face IS exposed
          top= 87 bot= 81 next side=263    Face IS exposed
          top=268 bot=... next side=...    Face IS exposed


Structure has   5 lines:
          Top is not connected.
          top=183 bot=269 next side=186    Face IS exposed
          top=188 bot=... next side=...    Face is not exposed
          top=151 bot=... next side=...    Face is not exposed


Structure has   7 lines:
          Top is not connected.
          top=... bot=... next side= 65    Face is not exposed
          top= 64 bot= 67 next side=270    Face IS exposed
          top= 63 bot=... next side=...    Face IS exposed
          top= 47 bot=... next side= 48    Face is not exposed


Structure has   7 lines:
          Top is not connected.
          top= 67 bot=... next side=125    Face is not exposed
          top=124 bot=272 next side=271    Face IS exposed
          top=273 bot=274 next side=...    Face IS exposed


Structure has   6 lines:
          Top IS connected.
          top=278 bot=... next side=...    Face is not exposed
          top= 64 bot=... next side=276    Face is not exposed
          top= 65 bot=277 next side=279    Face IS exposed


Structure has   9 lines:
          Top IS connected.
          top= 75 bot=281 next side=280    Face IS exposed
          top= 76 bot=284 next side=283    Face IS exposed
          top=286 bot=... next side=...    Face is not exposed
          top=285 bot=... next side=282    Face is not exposed
```

Figure 4-11c: Complex Building

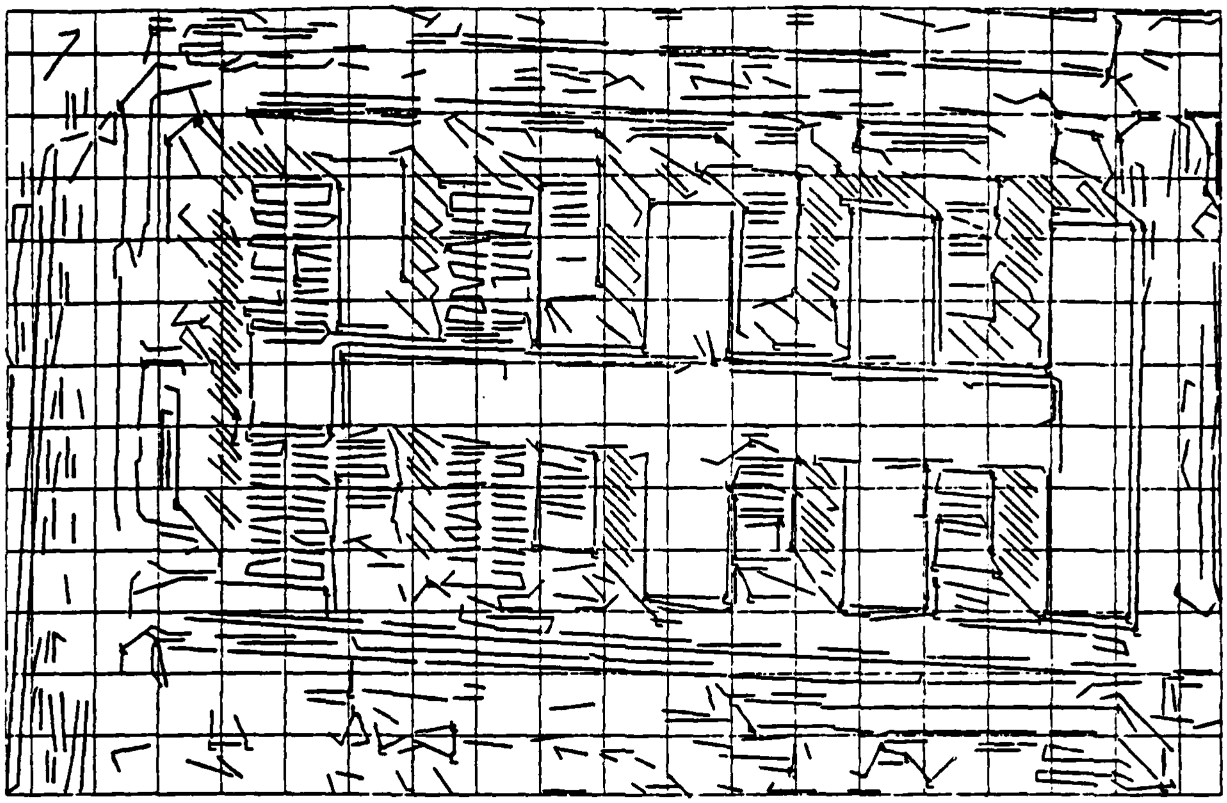
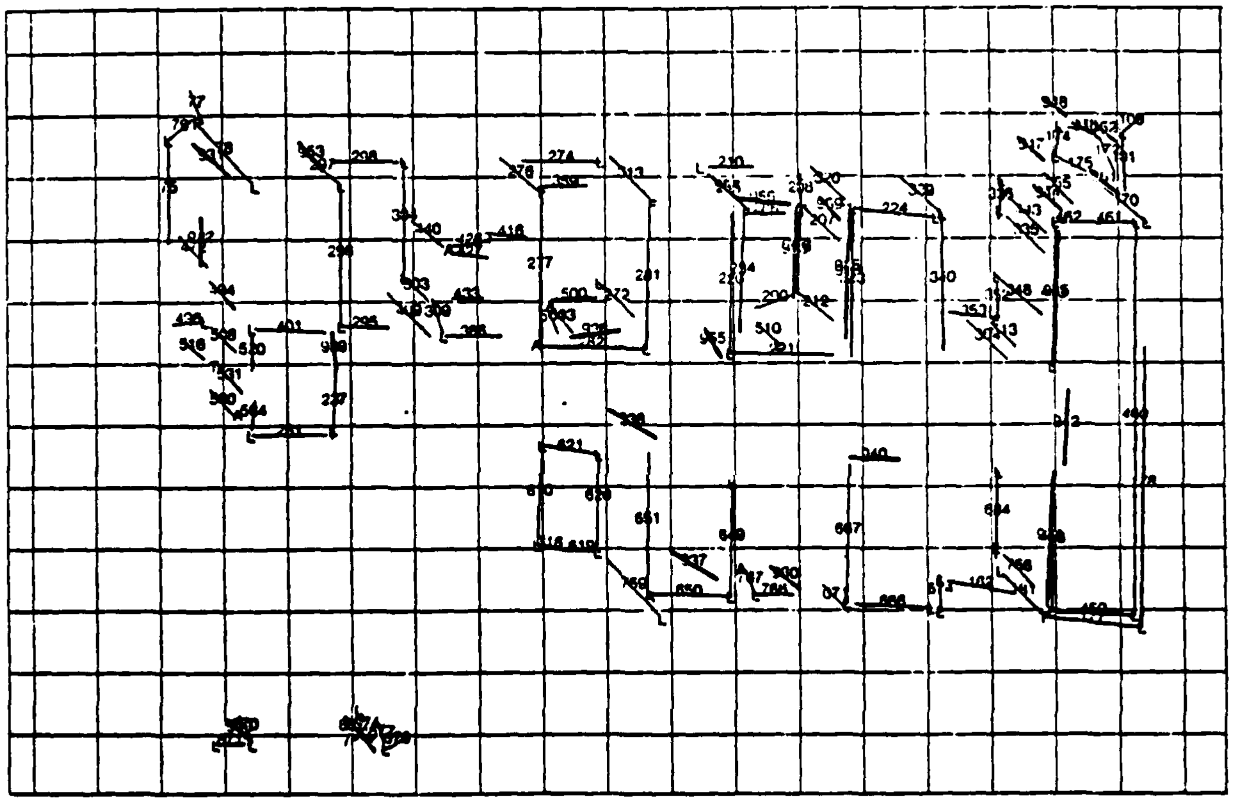


Figure 4-12a: Magnified Version of Complex Building


Fiqure 4-12b: Magnified Version of Complex Building

```
---------- ----

Structure has   6 lines:
        Top IS connected.
        top=618 bot=... next side=...   Face is not exposed
        top=619 bot=... next side=...   Face is not exposed
        top=620 bot=... next side=...   Face is not exposed
        top=621 bot=... next side=...   Face IS exposed
        top=610 bot=... next side=...   Face IS exposed

Structure has   7 lines:
        Top is not connected.
        top=... bot= 77 next side= 78   Face IS exposed
        top=... bot= 76 next side=931   Face IS exposed
        top=932 bot= 78 next side=472   Face IS exposed

Structure has   7 lines:
        Top IS connected.
        top=178 bot=... next side=...   Face is not exposed
        top=172 bot=... next side=948   Face is not exposed
        top=173 bot=... next side=948   Face IS exposed
        top=174 bot=... next side=947   Face IS exposed

Structure has   6 lines:
        Top is not connected.
        top=933 bot=... next side=...   Face is not exposed
        top=286 bot=210 next side=288   Face IS exposed
        top=284 bot=... next side=...   Face IS exposed

Structure has   4 lines:
        Top is not connected.
        top=178 bot=... next side=...   Face IS exposed
        top=177 bot=... next side=181   Face IS exposed
        top=934 bot=... next side=...   Face is not exposed

Structure has   6 lines:
        Top is not connected.
        top=... bot=... next side=207   Face is not exposed
        top=935 bot=213 next side=212   Face IS exposed
        top=... bot=290 next side=610   Face IS exposed

Structure has   5 lines:
        Top is not connected.
        top=... bot=... next side=348   Face is not exposed
        top=... bot=352 next side=613   Face IS exposed
        top=... bot=353 next side=354   Face IS exposed

Structure has   4 lines:
        Top is not connected.
        top=416 bot=... next side=...   Face IS exposed
        top=428 bot=... next side=440   Face IS exposed
        top=427 bot=... next side=...   Face is not exposed

Structure has   6 lines:
        Top is not connected.
        top=... bot=... next side=272   Face is not exposed
        top=936 bot=500 next side=503   Face IS exposed
        top=... bot=504 next side=...   Face IS exposed

Structure has   6 lines:
        Top is not connected.
        top=340 bot=... next side=339   Face is not exposed
        top=224 bot=... next side=320   Face IS exposed
        top=223 bot=208 next side=...   Face IS exposed

Structure has   6 lines:
        Top is not connected.
        top=649 bot=... next side=937   Face IS exposed
        top=660 bot=... next side=759   Face IS exposed
        top=661 bot=... next side=938   Face is not exposed

Structure has   7 lines:
        Top is not connected.
        top=359 bot=274 next side=276   Face IS exposed
        top=277 bot=... next side=...   Face IS exposed
        top=282 bot=... next side=...   Face is not exposed
        top=281 bot=... next side=313   Face is not exposed

Structure has   7 lines:
        Top is not connected.
        top=939 bot=... next side=...   Face is not exposed
        top=401 bot=... next side=494   Face IS exposed
        top=626 bot=... next side=508   Face IS exposed
        top=... bot=406 next side=618   Face IS exposed

Structure has   6 lines:
        Top is not connected.
        top=433 bot=... next side=303   Face IS exposed
        top=309 bot=... next side=499   Face IS exposed
        top=386 bot=... next side=...   Face is not exposed

Structure has   5 lines:
        Top is not connected.
        top=... bot=... next side=631   Face is not exposed
        top=564 bot=... next side=660   Face IS exposed
        top=261 bot=... next side=...   Face is not exposed
        top=237 bot=... next side=...   Face is not exposed

Structure has   6 lines:
        Top is not connected.
        top=940 bot=... next side=...   Face IS exposed
        top=667 bot=... next side=671   Face IS exposed
        top=666 bot=... next side=...   Face is not exposed
        top=674 bot=... next side=...   Face is not exposed
        top=182 bot=... next side=...   Face is not exposed

Structure has   3 lines:
        Top is not connected.
        top=... bot=... next side=297   Face is not exposed
        top=296 bot=... next side=...   Face IS exposed
        top=296 bot=... next side=...   Face is not exposed

Structure has   9 lines:
        Top is not connected.
        top=942 bot=... next side=...   Face IS exposed
        top=458 bot=684 next side=788   Face IS exposed
        top=459 bot=... next side=...   Face is not exposed
        top=460 bot=... next side=170   Face is not exposed
        top=461 bot=... next side=366   Face IS exposed

Structure has   7 lines:
        Top IS connected.
        top=463 bot=336 next side=335   Face IS exposed
        top=946 bot=... next side=944   Face is not exposed
        top=462 bot=... next side=443   Face IS exposed

Structure has   3 lines:
        Top is not connected.

        top=870 bot=... next side=949   Face IS exposed
        top=871 bot=... next side=...   Face IS exposed

Structure has   3 lines:
        Top is not connected.
        top=877 bot=... next side=960   Face IS exposed
        top=876 bot=... next side=...   Face is not exposed

Structure has   4 lines:
        Top is not connected.
        top=100 bot=... next side=962   Face IS exposed
        top=101 bot=... next side=961   Face IS exposed

Structure has   3 lines:
        Top is not connected.
        top=304 bot=... next side=...   Face is not exposed
        top=298 bot=... next side=963   Face IS exposed

Structure has   8 lines:
        Top is not connected.
        top=957 bot=... next side=...   Face is not exposed
        top=956 bot=... next side=964   Face IS exposed
        top=220 bot=... next side=966   Face IS exposed
        top=221 bot=... next side=...   Face is not exposed
        top=958 bot=... next side=969   Face is not exposed

Structure has   3 lines:
        Top is not connected.
        top=... bot=... next side=980   Face is not exposed
        top=786 bot=... next side=...   Face IS exposed
        top=787 bot=... next side=...   Face is not exposed

Structure has   3 lines:
        Top is not connected.
        top=868 bot=... next side=...   Face is not exposed
        top=874 bot=... next side=982   Face IS exposed
```

Figure 4-12c: Magnified Version of Complex Building

4.4 Access Efficiency of Line Features by Image Sectoring

4.4.1 Introduction

A gray scale image is processed through various steps to obtain a list of line segments which represent the edges in the image. To perform further processing on the image it is desireable to access these lines in an efficient and yet relatively simple manner. The principle accessing operation is windowing, finding all lines which pass through a specified area, the window. There are two general ways to do this. The first approach is to search all the lines in the image and test if each passes in the specified window. Storing the lines in order by the x,y coordinates of their endpoints would limit the search. However, when the image contains a significant number of lines this approach becomes slow. The second approach is to divide the image into a number of smaller areas called sectors. The image bounds of each sector and the lines passir in it are stored. A window operation requires two separate tasks. First, finding which sector(s) contain the window, and then testing only the lines within the required sectors if they are in the window. The time to find which sectors to search is a function of the number of sectors. The time searching within the sectors is a function of both the number of sectors and the size of the window. The sum of these two times plus a small amount of overhead is the total access time. There should be an optimum number of sectors to divide the image into so that the access time is minimized.

4.4.2 Implementation

A 'C' language program was written to obtain the necessary timing data. The program does a square window operation at the endpoints of each line in the image. This is similar to what a program to form junctions of lines would do. A subroutine performs the window operation and returns a list of lines within the window. For a single run on the lines in an image the program provides the following data:

1. Total CPU time for all the calls to the windowing subroutine.
2. Average time for each call to the subroutine determining which sectors contain the current window.
3. Total number of sectors searched for all the calls to the subroutine.
4. Average time spent searching within a sector.
5. Total time searching within sectors. The product of 3 and 4 above.
6. Overhead. The difference of 2 and 5 from 1.

4.4.3 Results

The program was run on a 150 by 150 pixel image. The image is divided into N rows and N columns of sectors. The number of sectors, N, was varied from 1 to 30. Square windows of width 3, 5, 10, 15, 25 and 30 pixels were used. Each trial (for a particular number of sectors and window width) was iterated three times and the results averaged. The program was run late at night (2 AM+) to avoid fluctuations in the computer's load. The results are plotted in the graph in Fig. 4-13. The line image used in the test is shown in Fig. 4-14.

It is evident from the graph that the optimum number of sectors to use is from N=6 to N=8. The number does not depend on the size of the window.

4.4.4 Summary

The line features within an image need to be easily and quickly accessed to perform higher level processing of the image. The line features of an image can be stored in a single list, but this requires that the entire list be scanned during a search. Improved access time can be achieved by dividing the image area into a number of smaller areas called sectors. Each sector contains a list of only the line features in its area. The same search now requires that only the lists of particular sectors be scanned. The access efficiency is directly related to the number of sectors used. A test program was written to determine the variation of access time with respect to the number of sectors an image is divided into. The fastest access was obtained when the image was divided into sectored areas forming from 6 to 8 rows and columns.

4.5 Representation and Incremental Construction of a Three-Dimensional Scene Model

4.5.1 Introduction

The representation, construction, and updating of a 3D scene model is described. The scene model is a surface-based description of an urban scene, and is incrementally acquired from a sequence of images obtained from multiple viewpoints. We assume here that from each view of the scene, a 3D wire-frame description that represents portions of edges and vertices of buildings can be extracted.

The central scene model is a surface-based description which is constructed and modified from these features. It is represented as a graph in terms of primitives such as faces, edges, vertices, and their topology and geometry. It also has mechanisms to add and delete hypotheses for parts of the scene for which there are partial data. Before modifications to the scene model can occur, the 3D features from the new view must be matched to the current model. The scene model may, at any point along its development, be used for tasks such as image interpretation, planning, or display generation. A new view may then be acquired which may further modify the model.

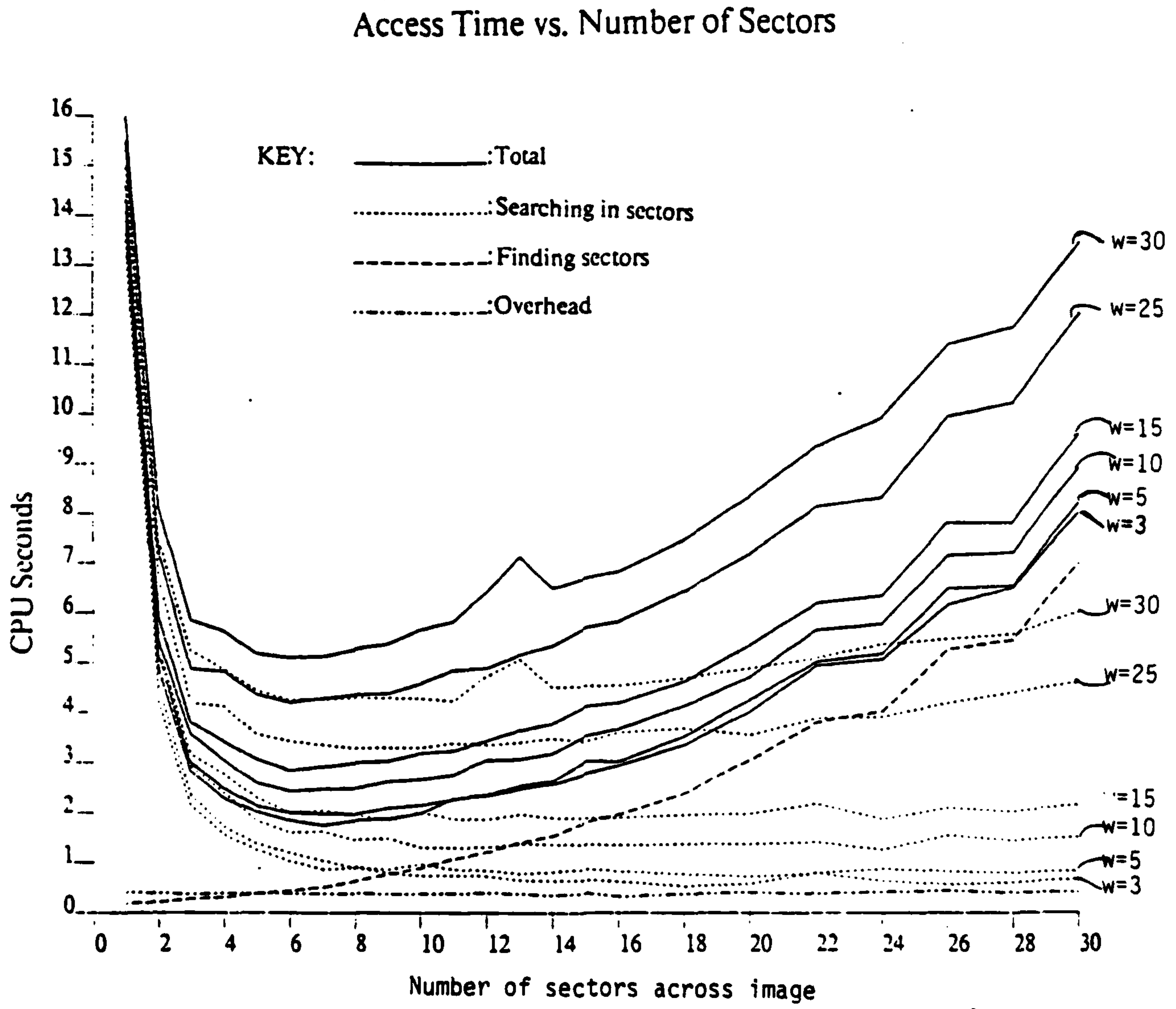


Figure 4-13: Access time for different numbers of sectors.

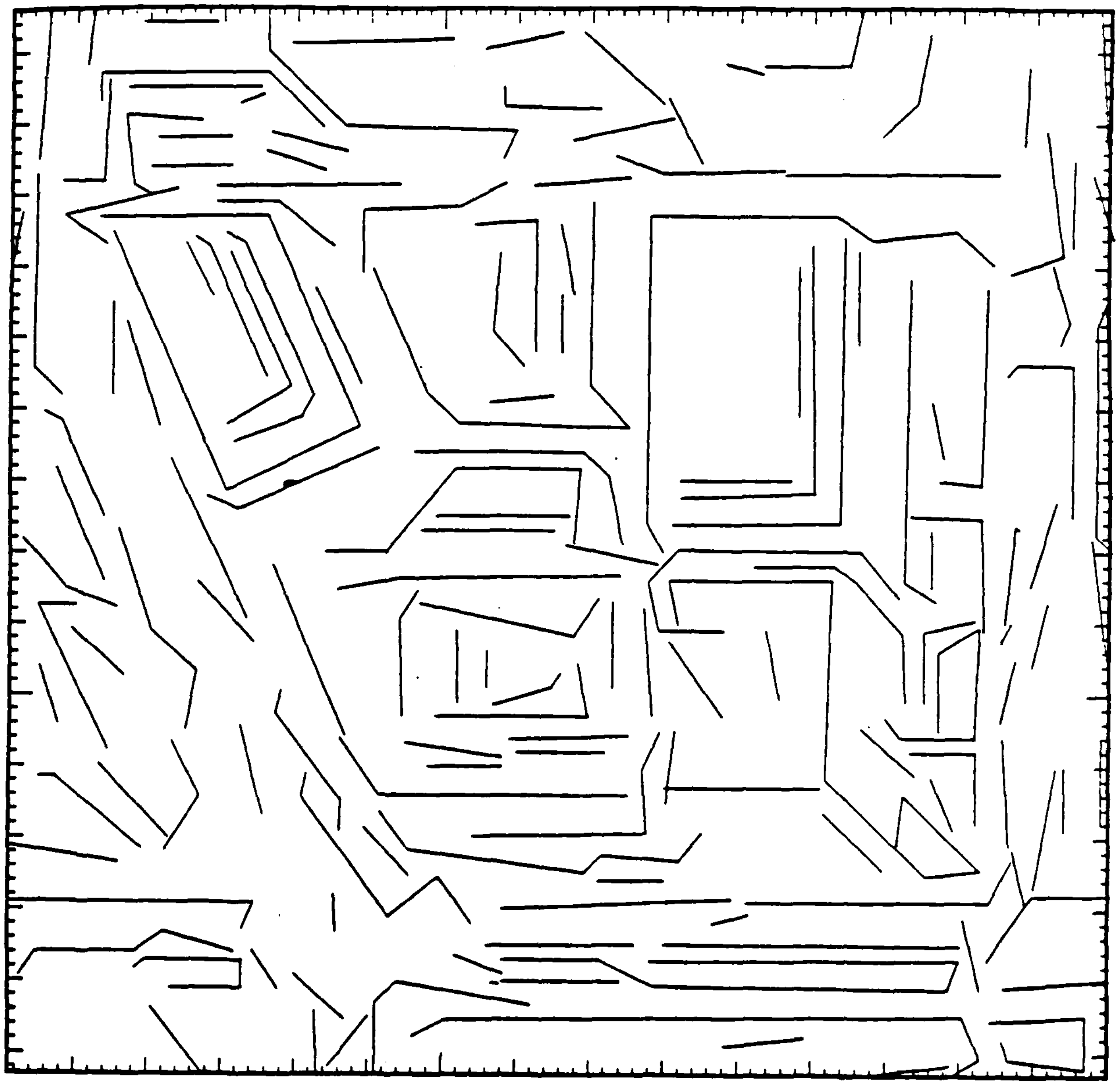


Figure 4-14: Line image used for sector timing

4.5.2 Representing and Manipulating the 3D Scene Model

The representation we have developed for the 3D scene model draws on ideas from geometric modelling used in computer-aided design systems [Baer, Eastman, and Henrion 79, Requicha 80]. In these systems, however, the 3D models are usually derived through interaction with a user. Our case is different in that (1) the 3D models are to be derived automatically from 2D images, and (2) many portions of the scene will be unknown or recovered with errors because of occlusions or unreliable analysis.

The following factors have determined how the scene model is represented and manipulated.

1. Partially complete, planar-faced objects must be efficiently described by the model. It is therefore represented as a graph in terms of symbolic primitives such as faces, edges, vertices, and their topology and geometry. Information is added and deleted by means of these primitives.

2. The model must be easy to use in matching.

3. Because scene approximations are often more useful if they contain reasonable hypotheses for parts of the scene for which there are partial data, we introduce mechanisms that permit hypotheses to be generated, added, and deleted.

4. Because incremental modifications to the model must be easy to perform, we introduce mechanisms to (a) add primitives to the model in a manner such that constraints on geometry imposed by these additions are propagated throughout the model, and (b) modify and delete primitives if discrepancies arise between newly derived and current information.

4.5.2.1 Representation of Model

The 3D structure in the scene is represented in the form of a graph, called the *structure graph.* The nodes and links represent primitive topological and geometric constraints. The structure graph is incrementally constructed through the addition and deletion of these constraints. As constraints are accumulated, their effects are propagated to other parts of the graph so as to obtain globally consistent interpretations.

The current structure-graph representation models surfaces in the scene as polyhedra. The components of a polyhedral surface are the face, edge, and vertex. We distinguish the topology of the polyhedral components from their geometry [Baer, Eastman, and Henrion 79, Eastman and Preiss 82]. The geometry involves the physical dimensions and location in 3-space of each component, while the topology involves connections between the components.

4.5.2.2 Primitive Constraints

In the structure graph, nodes represent either primitive topological or primitive geometric elements. We define five types of primitive topological elements: faces, edges, vertices, objects, and edge-groups. The first three are components of the polyhedral surface. The last two are introduced in order to conveniently represent connected groups of elements. The object is intended to represent a connected set of faces that enclose a volume in 3-space. The edge-group is intended to represent a connected ring of edges that enclose an area in 2-space on a face. Because of the partial nature of the structure graph, however, an object may represent any set of faces, edge-groups, edges, and vertices that are potentially part of a single, closed, connected unit. Similarly, an edge-group may represent any set of edges and vertices that are potentially part of a single edge ring on a face.

Face, edge, and vertex nodes are tagged as either *confirmed* or *unconfirmed.* Confirmed means that the element represented by the node has been derived directly from images. Unconfirmed means that the element has only been hypothesized.

We define three types of primitive geometric elements: planes, lines, and points. These serve to constrain the 3-space locations of faces, edges, and vertices. Plane and line nodes contain plane and line equations, respectively. Point nodes contain coordinate values.

Although an edge is ideally delimited by two vertices, edges derived from images are often incomplete and may be delimited by one vertex and an end point which is not necessarily a vertex. Such a point is tagged as an end point. A point may also be tagged as confirmed or unconfirmed, depending on whether or not it has been derived from images. This is useful when a confirmed edge is hypothesized to extend further in length. The confirmed portion of the edge lies between confirmed points, while the unconfirmed portion may be delimited on one side by an unconfirmed point.

The structure graph contains two types of links: the *part-of* link, representing the part/whole relation between two topological nodes, and the *geometric constraint* link, representing the constraint relation between a geometric and topological node. For example, a vertex may be part of an edge, edge-group, face, or object. A point constrains the position of a vertex, edge, or face if it lies on the vertex, edge, or face, respectively.

Although several points may be constrained to lie on, say, a face, the points need not necessarily be coplanar. The equations of planes for faces and lines for edges are currently obtained by a least squares fit to all the points constraining the face or edge. In general, therefore, edges and vertices that are part of a face need not lie on the plane of the face, and vertices of an edge need not lie on the line of the edge.

Fig. 4-15 shows a simple example of a structure graph consisting of two objects, *ob1* and *ob2.* Arrows with single lines represent part-of links, and arrows with double lines represent geometric constraint links. The object *ob1* contains ten faces, *f1* ... *f10.* The faces *f1* and *f2* each contains one edge-group, *g1* and *g2,* respectively. Notice that *f10* contains two edge-groups. These might represent a bounding outer ring of edges and an inner ring of edges that bounds a hole in the face. The edge-groups contain edges, each of which may

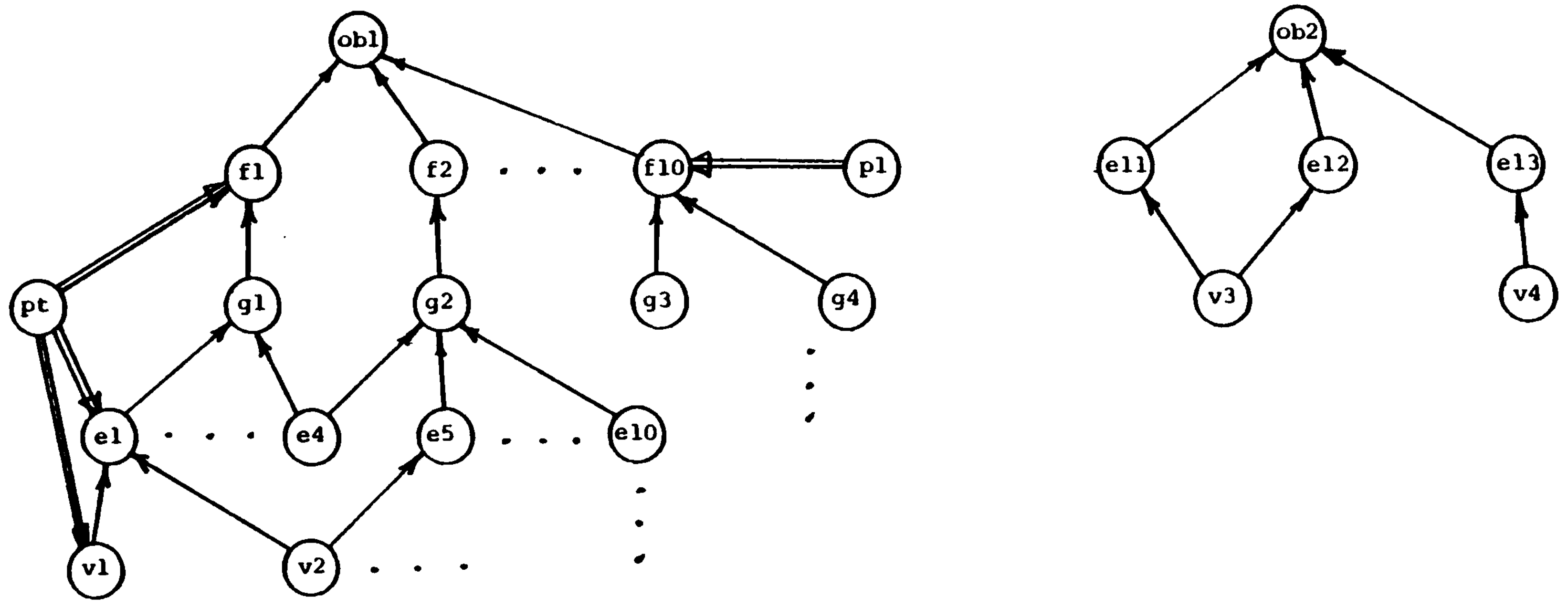
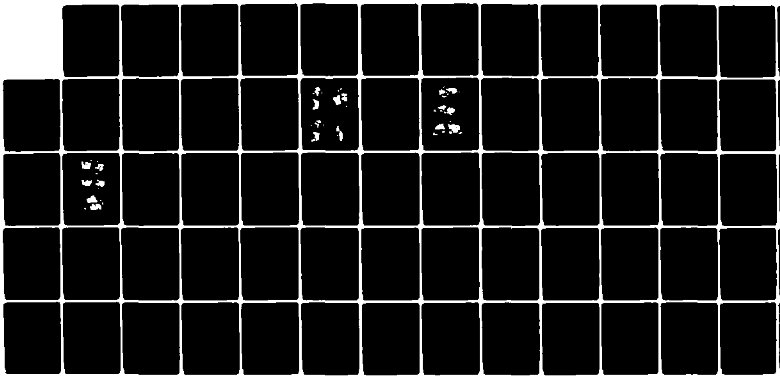


Figure 4-15: Simple example of a structure graph consisting of two objects, *ob1* and *ob2*. Double line arrows represent geometric constraint links, and single line arrows represent part-of links.

AD-A144 357 MULTI-DISCIPLINARY TECHNIQUES FOR UNDERSTANDING TIME-VARYING SPACE-BASED..(U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF ELECTRICAL ENGINEE..

UNCLASSIFIED D CASASENT ET AL. 01 JUN 84 F/G 20/6 NL

2/2

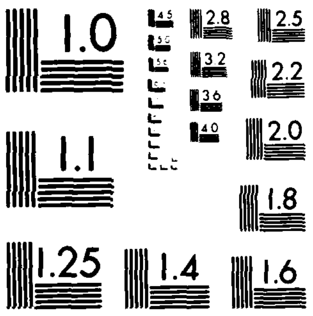


END

DATE FILMED

9-84

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

be part of more than one edge-group. The edges, in turn, contain vertices. The point *pt* constrains *v1*, *e1*, and *f1*. The plane *pl* constrains *f10*. The other object in the structure graph, *ob2*, is highly incomplete. It contains only three edges and two vertices.

4.5.3 Modifications to the 3D Scene Model

Modifications to the structure graph are made by adding or deleting nodes and links, or changing the equations of line and plane nodes, or the coordinates of point nodes. All effects of modifications are propagated to other parts of the graph.

4.5.3.1 Propagation Due to Geometric Modifications

Consider adding or deleting a geometric constraint link between a geometric and topological node. Any of the three geometric nodes (points, lines, and planes) may constrain any of the three topological nodes (vertices, edges, and faces). Object and edge-group nodes may not be geometrically constrained directly. Fig. 4-16 shows how a constraint on one node may propagate to others. The arrows in the figure indicate the direction of propagation. The tail of an arrow indicates the source constraint; the head indicates the constraint implied by the source constraint.

We see in Fig. 4-16 that point constraints propagate upward. That is, if a point constrains a vertex, it must also constrain all edges and faces which contain that vertex. Similarly, a point that constrains an edge must also constrain all faces containing that edge. Note that when a point constrains an edge, we assume that no constraint is implied for arbitrary vertices that are part of that edge, since the point need not lie on any of these vertices. In one sense, the point may be considered to constrain such vertices since they must lie on a line going through the point. This constraint, however, is not useful until another constraint on the line is derived, such as another point that lies on the edge. In this case, our system generates the equation of the line that constrains the edge and propagates the line constraint down to the vertex, as explained in the next paragraph. A more direct and useful constraint is thus imposed on the vertex. Similarly, when a point constrains a face, no useful constraint is implied for arbitrary edges or vertices that are part of the face.

As indicated in Fig. 4-16, line constraints propagate outward. A line that constrains an edge must also constrain all faces containing the edge and all vertices that are part of the edge. Finally, plane constraints propagate downward. A plane that constrains a face must also constrain all edges and vertices that are part of the face. Similarly, a plane that constrains an edge must also constrain all vertices that are part of the edge. Whenever a geometric constraint link is added, propagation occurs as indicated in Fig. 4-16.

When a geometric constraint link is deleted, the rest of the structure graph must be made consistent with this change. Our approach to this problem is based on the TMS system [Doyle 79], using the notion that

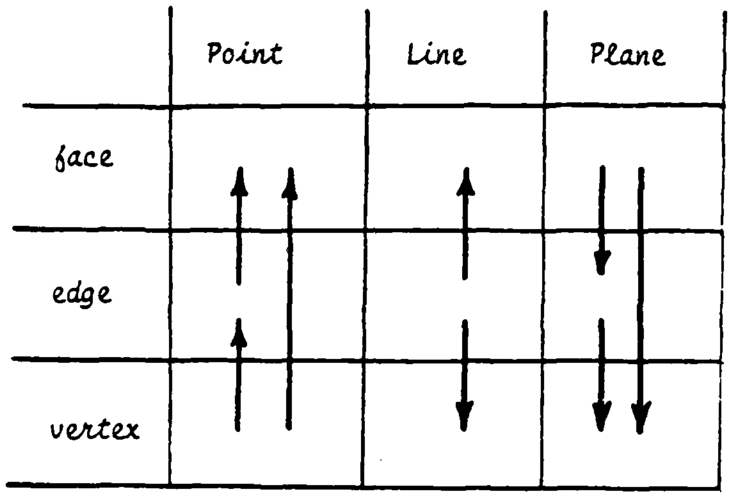


Figure 4-16: Rectangular boxes indicate geometric constraints on topological nodes. Arrows indicate direction of propagation of constraints.

when an assertion is deleted, all assertions implying it and all assertions implied by it that have no other support should also be deleted. To see this, consider Fig. 4-17. Let $\{x_1, x_2, \ldots, x_m\}$ be a set of assertions, each of which independently implies the assertion y. The assertion $(y \wedge v_1 \wedge v_2 \wedge \ldots)$, in turn, implies each assertion in the set $\{z_1, z_2, \ldots, z_n\}$. Furthermore, for each i, z_i is independently implied by each assertion in the set $\{w_{ij}\}$. Now suppose the assertion y is deleted, i.e., it is declared false. Then

1. Since each assertion z_i depends on the truth of y, z_i is deleted unless it has other support w_{ij}.
2. All assertions x_i are made false. None of them can be true, for if one were, y must be true. Since x_i may consist of a conjunction of assertions, at least one of them is deleted to make x_i false.

We obtain assertions that imply a given assertion by following backwards along the arrows in Fig. 4-16, and we obtain assertions implied by a given assertion by following forward along the arrows.

Consider the simple example in Fig. 4-18a, which depicts three topological nodes (vertex v, edge e, face f) constrained by one geometric node (point p). Suppose now that link 4 is deleted (Fig. 4-18b), that is, the assertion "p constrains e" is deleted. All assertions which have implied this must now be deleted, for if one were to hold, link 4 would also hold. To find these assertions, we locate the box in Fig. 4-16 that represents a point constraining an edge and follow backwards along the arrow. The result is the box that represents the point constraining any vertex of the edge. In Fig. 4-18b, this corresponds to the assertion "p constrains v, and v is part of e". This assertion must therefore be made false. To do so, we may delete either link 1, link 3, or both from Fig. 4-18b. Our intuition tells us that part-of links (link 1) should dominate constraint links (link 3), and thus link 3 is deleted. This seems to work well for our examples.

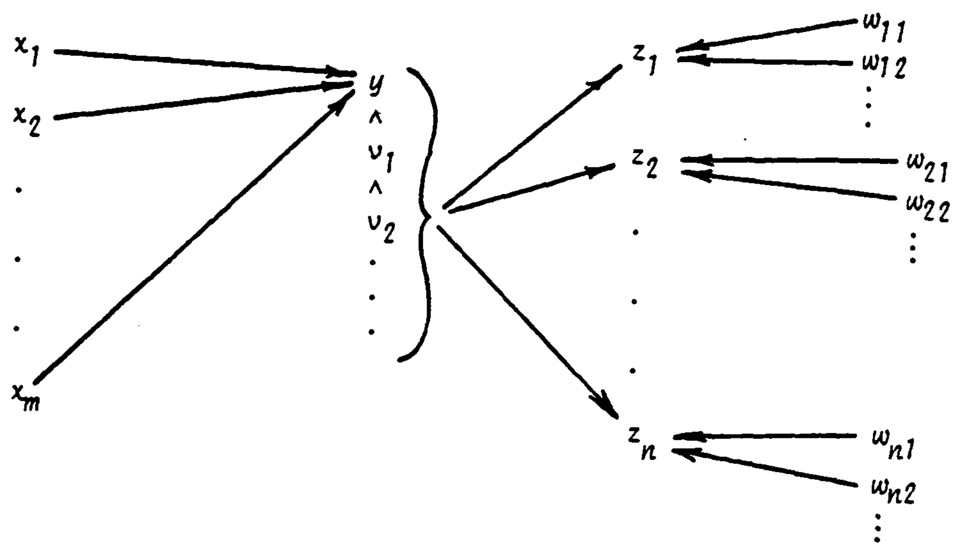


Figure 4-17: The assertion y is independently implied by each x_i. Each assertion z_i is independently implied by $(y \wedge v_1 \wedge v_2 \wedge \ldots)$ and w_{ij}

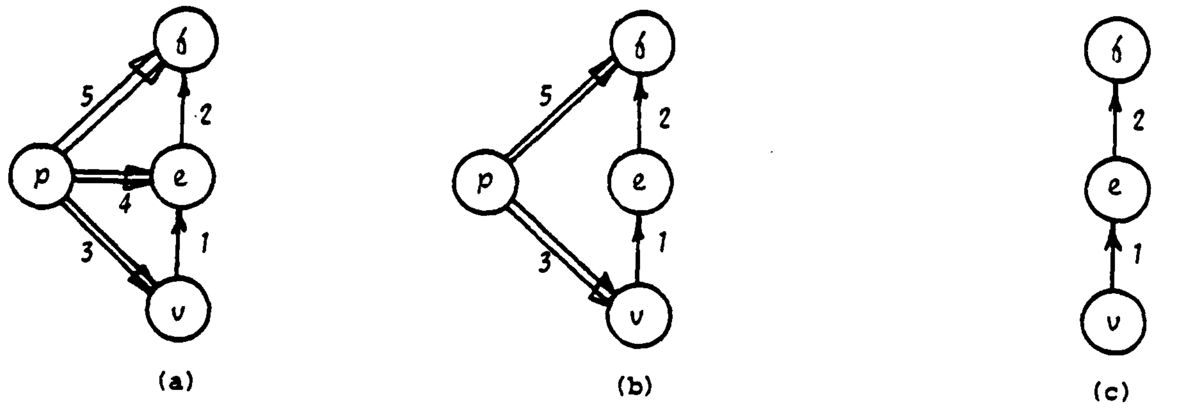


Figure 4-18: (a) Initial structure graph. (b) Link 4 is deleted. (c) Resulting structure graph after effects of deletion have been propagated.

We now must determine the assertions implied by the one initially deleted. All these assertions must also be deleted unless they have other support. To do so, we follow forward along the arrow from the box in Fig. 4-16 that represents a point constraining an edge, and the result is the box that represents the point constraining all faces containing the edge. In Fig. 4-18b, this corresponds to the assertion "*p* constrains *f*", which is link 5. This link should therefore be deleted since it has no other support. One possible source of other support is external to the structure graph. Link 5 may have been derived, for example, directly from image data, rather than through structure graph propagation. We rule out the possibility that links 4 and 5 are unrelated, and thus delete link 5. The resulting structure graph is depicted in Fig. 4-18c.

4.5.3.2 Propagation Due to Topological Modifications

When a topological part-of link between two topological nodes is added or deleted, the effects are propagated to other parts of the structure graph. In the following, we will consider both geometric and topological effects.

4.5.3.3 Geometric Effects

When a topological part-of link is added between two topological nodes, the geometric constraints on each node must be propagated to the other node in accordance with the chart in Fig. 4-16. There are three main cases to consider: (1) adding a part-of link between a vertex and edge node, (2) between an edge and face node, and (3) between a vertex and face node. These three cases are explicitly covered in Fig. 4-16. The remaining cases fall into two classes: (a) adding a part-of link between some topological node and an object node, and (b) between some topological node and an edge-group node. Since object nodes cannot be geometrically constrained directly, actions in class (a) have no geometric effects. Since geometric constraints can be propagated through edge-group nodes, actions in class (b) do have geometric effects. These effects, however, can be reduced to the three cases above, as explained in the next paragraph.

Consider the example of adding a part-of link between an edge node E and a face node F. From Fig. 4-16, we see that all point and line constraints on E must be propagated to F, while all plane constraints on F must be propagated to E. Plane constraints propagated to E are, in turn, propagated to vertices of E. As another example, consider adding a part-of link between an edge-group node G and a face node F. This situation results in the same geometric propagation as the following two cases: (1) add a part-of link from each edge of G to F, and (2) from each vertex of G to F. Similar rules can be established for the other two situations involving edge-group nodes (i.e., adding a link between a vertex and edge-group node, and between an edge and edge-group node).

When a part-of link between two topological nodes is deleted, an attempt is made to nullify any geometric propagation that occurred through the link. This is done by deleting, from the two nodes connected by the link, all geometric constraints that have propagated through the link. The effects of deleting these geometric constraint links are, in turn, propagated to the rest of the graph in the manner described in the previous section.

As an example, consider deleting a part-of link between an edge node E and a face node F. As seen in Fig. 4-16, all point and line constraints on F that also constrain E were either (1) propagated up from E, (2) propagated up from another edge or vertex of F, or (3) derived from an external source. We rule out the possibility that the same constraints on E and F are unrelated, thus ruling out the external source. Therefore, points and lines that constrain both F and E, but do not also constrain another edge or vertex of F, are deleted from F since we just cut off the only path through which they could have propagated to F. The effects of deleting the point and line constraints from F are, in turn, propagated to the rest of the graph. Similarly, all plane constraints on E that also constrain F are deleted from E unless they also constrain another face that contains E (which would be unusual). The effects of deleting plane constraints from E are then propagated.

An example of a link with more than one source of support is shown in Fig. 4-19a. Suppose the part-of link between *e1* and *f*, link 4, is deleted (Fig. 4-19b). According to the chart in Fig. 4-16, link 8 is a candidate for deletion since the point node *p* constrains both *e1* and *f*. However, since *p* also constrains the edge *e2*, which is part of *f*, link 8 is still valid.

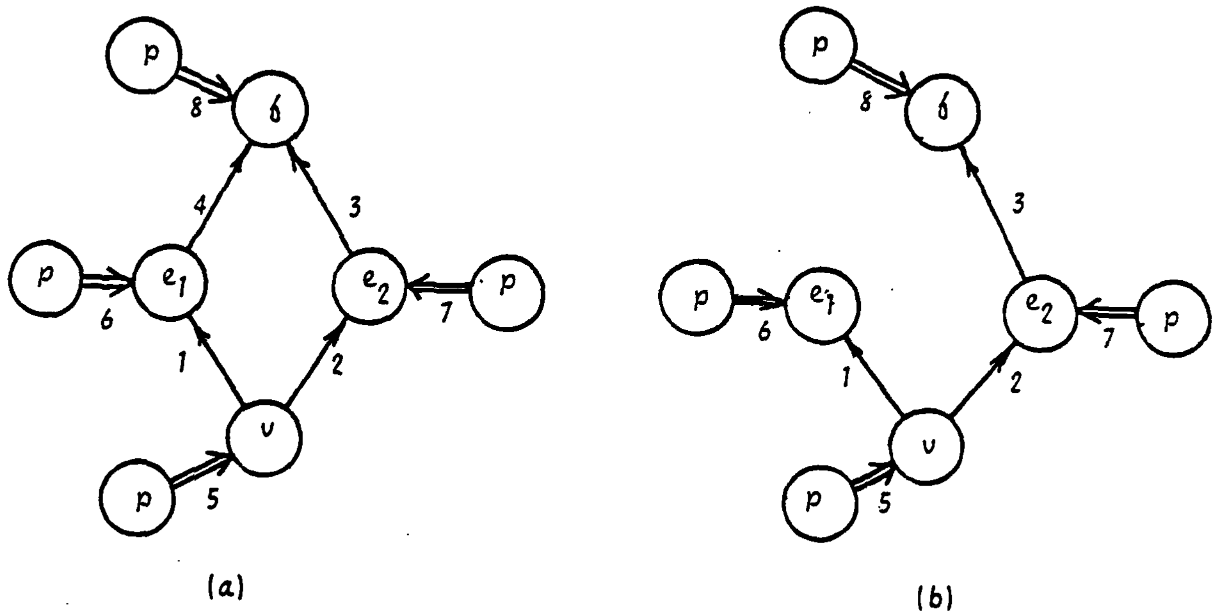


Figure 4-19: Example of a link with more than one source of support. (a) Initial structure graph. (b) Link 4 is deleted, but link 8 remains because of support from links 3 and 7.

4.5.3.4 Topological Effects

A topological modification sometimes implies topological changes elsewhere in the structure graph. This is best illustrated through an example. Fig. 4-20a shows the graph representing the situation in Fig. 4-20b. The edge *e* has two vertices, *v1* and *v2*, and *v1* is known to be part of the face *f*. Now suppose a part-of link is added between *v2* and *f* (link 4 in Fig. 4-20c). Since both vertices of *e* are now part of *f*, *e* must also be part of *f*, as shown in Fig. 4-20d. Therefore link 5 in Fig. 4-20c is added.

Another kind of topological effect results from the desire to eliminate redundant part-of links. Part-of links serve as paths in the structure graph along which effects of geometric changes are propagated. In order to simplify this process, the number of paths between each pair of topological nodes is minimized using the following rule: Two topological nodes may not be directly connected (i.e., by means of a part-of link) if they are also connected through one or more intermediate topological nodes. For example, suppose a part-of link is added between the edge node *e* and the face node *f* in Fig. 4-21a. To avoid redundancy, all links connecting vertex nodes of *e* and the node *f* (link 1 in Fig. 4-21a) and vertex nodes of *e* and object nodes containing *f* (link 2) are deleted. In addition, if there were any links between *e* and object nodes containing *f*, they would also be deleted. The final configuration is shown in Fig. 4-21b. In the example of Fig. 4-20, the graph in (c) has redundant links. Links 1 and 4 are therefore deleted, resulting in the graph of Fig. 4-20e.

Although adding a part-of link can result in topological changes elsewhere in the graph, deleting a part-of link does not change the topology anywhere else. No attempt is made to recover previous states of topological connections. Fig. 4-22b shows the result of deleting link 1 from the graph in Fig. 4-22a. This technique seems to work well in our experiments.

4.6 Constructing and Updating the 3D Scene Model

Each view of the scene undergoes analysis which results in a 3D wire-frame description representing 3D vertices and edges corresponding to portions of boundaries of objects in the scene. The goal of the updating process is to merge the wire-frame description with the current model. In general, this process will result in a partial 3D model which may consist of surfaces at some places but only portions of boundaries at other places. This partial 3D model must then be converted into a full surface-based description by hypothesizing new vertices, edges, and faces. Our current techniques for making such hypotheses exploit task-specific knowledge that falls into two categories: (1) knowledge of planar-faced objects, and (2) knowledge of urban scenes. These categories will be explored in detail in the next two sections.

Both the wire frames and scene models are represented by structure graphs. The wire-frame description extracted from the first view forms the initial state of the scene model, and all of its edges, vertices, and points

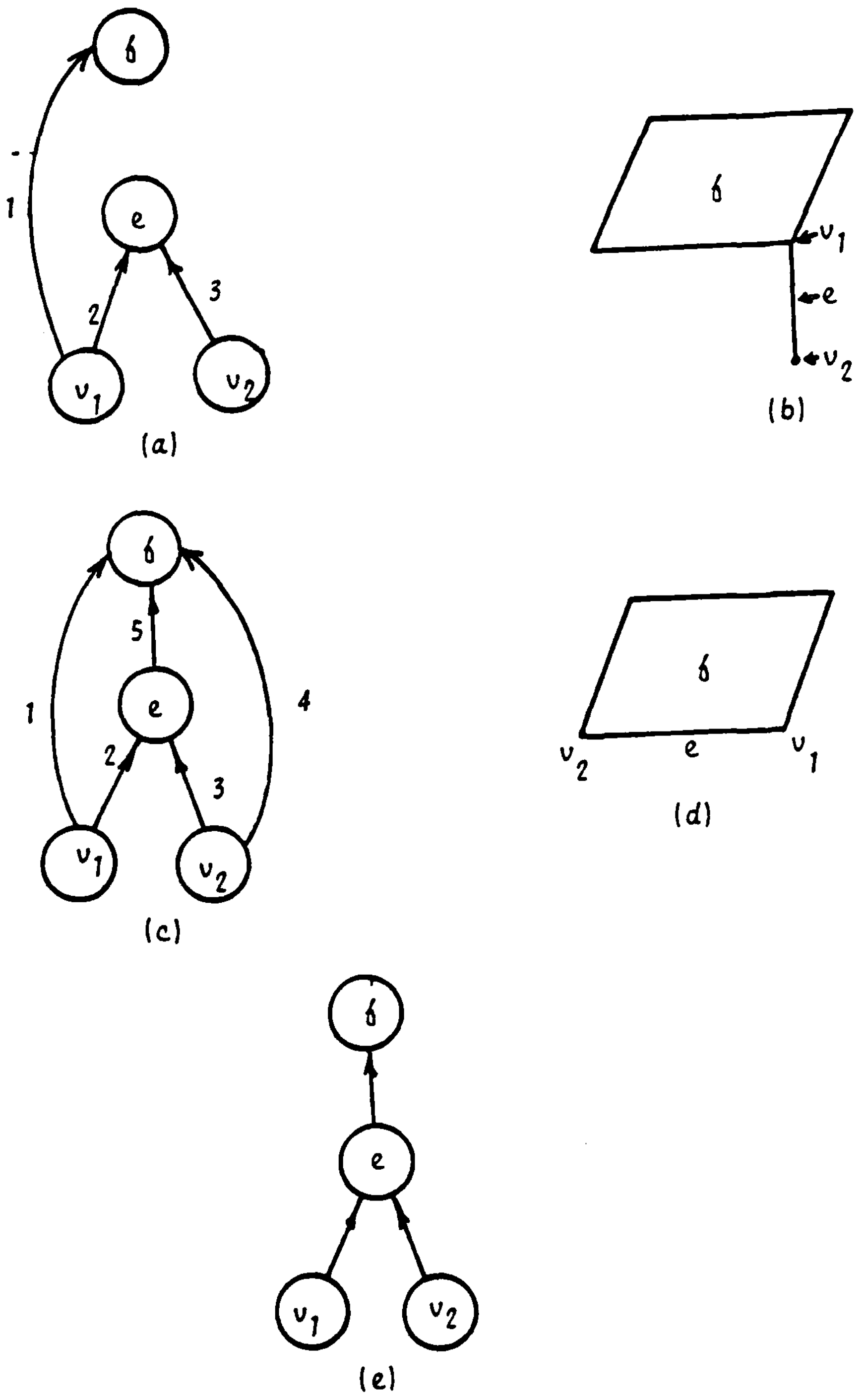


Figure 4-20: Topological propagation. (a) and (b) Initial situation. (c) and (d) Link 4 is added, resulting in addition of link 5. (e) Redundant links are eliminated.

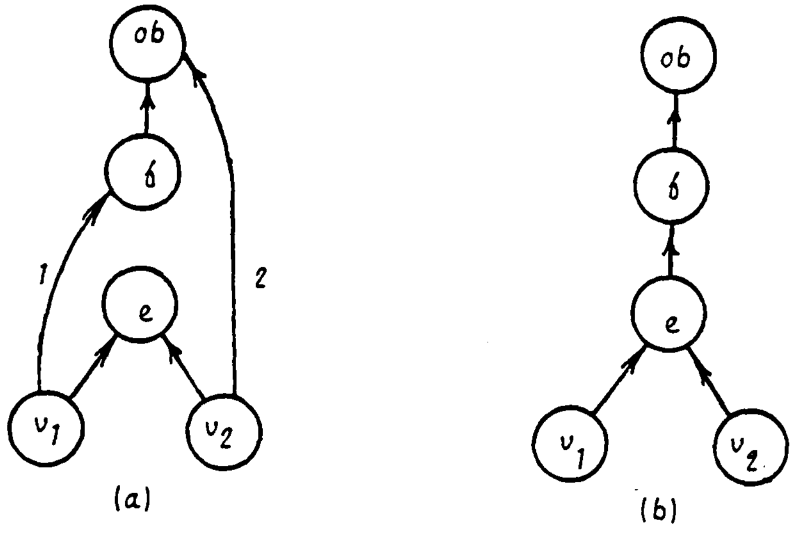


Figure 4-21: (a) Initial configuration. (b) When a link is added from *e* to *f*, links 1 and 2 are deleted to eliminate redundancy.

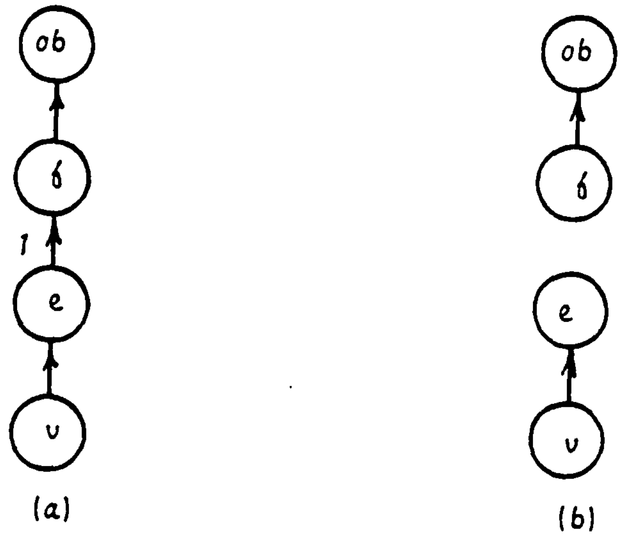


Figure 4-22: (a) Initial configuration. (b) Final result after link 1 is deleted.

are tagged as confirmed. This wire-frame model is then converted into a full surface-based model using task-specific knowledge. All elements of the model that were not present in the initial state are hypothesized and tagged as unconfirmed.

When a wire-frame description is extracted from a new view, all of its edges, vertices, and points are tagged as confirmed. This description is then matched to the current model (in order to find corresponding elements in the two and the coordinate transformation from one to the other) and merged with the current model. In the merging process, confirmed elements in the wire-frames and model that match are "averaged" together, resulting in new confirmed elements. Parts of the wire-frames that have no match in the model are then added to the model. Hypothesized elements in the model that are no longer consistent with confirmed parts are deleted. At this point, task-specific knowledge is again used to fill out the model and to form a full surface-based description.

4.6.1 Knowledge of Planar-Faced Objects

Since the structure graph has been designed for scenes that can be modelled as collections of planar-faced objects, knowledge of such objects is inherent in the representation and propagation rules, as described previously. In this section, we discuss how knowledge of such objects is used to construct a scene model from wire frames.

When new wire-frame information (derived either from the first or a subsequent view) is added to the model, many object descriptions will be incomplete. A goal of the model construction process, of course, is to complete these object descriptions using task-specific knowledge. The notion of an object description being complete is best expressed in the context of the structure graph. An object node in the structure graph is considered complete if it meets certain requirements, which may be expressed in terms of complete nodes contained by the object node. Each type of node in the graph, therefore, must meet certain requirements to be considered complete. Even though these requirements are only implicitly followed during the model construction process, it is useful to state them explicitly.

1. An object node is complete if it is closed, i.e., each edge node of the object is part of two face nodes, both of which are complete.

2. A face node is complete if it is constrained by a plane node and contains one or more complete edge-group nodes. One of these edge-group nodes must represent a bounding ring of edges on the face. The other, optional edge-group nodes represent inner edge rings, which would be holes in the face. In addition, each edge node of the face must be part of an edge-group of the face.

3. An edge group node is complete if it contains a single, connected, closed ring of complete edges on a face.

4. An edge node is complete if it is constrained by a line node and contains two complete vertex nodes.

5. A vertex node is complete if it is constrained by a point node.

In the following, we discuss heuristics applicable to planar-faced objects which are used in constructing the model.

4.6.1.1 Combining Edges

If there are two confirmed edges in the model that are nearly parallel, very close to each other, and overlap significantly, they are merged into a single edge, which is also labeled as confirmed. The test to determine parallelism and closeness involves checking whether all the points on one edge are within a threshold distance from the line constraining the other edge, and vice versa. The test for overlap involves projecting one edge onto the line of the other and measuring the amount of overlap.

In determining how to merge two such edges, we have thus far considered only one situation, depicted in Fig. 4-24a. Edges *e1* and *e3* satisfy the merging condition, and each has a single confirmed vertex (*v1* on *e1* and *v2* on *e2*). Furthermore, the confirmed vertices are on opposite ends of each other. This situation is handled by merging the two edges into a single edge whose two end points are the two confirmed vertices, as shown in Fig. 4-24b. This situation occurs only once in Fig. 4-23, for the two edges labeled E1 and E2.

4.6.1.2 Generating Web Faces

Each vertex in the model is assumed to correspond to a corner of an object. Therefore each adjacent pair of legs ordered around the vertex corresponds to the corner of a planar face. Thus far in our experiments, we have dealt only with trihedral vertices. In this case, every pair of legs of each vertex corresponds to the corner of a separate face. A partial face, called a *web face*, is generated for each pair.

Fig. 4-25a shows three web faces generated from a trihedral vertex. A web face may lie on either side of a vertex corner. In Fig. 4-25b, the web face is on the "inside", while in (c) it is on the "outside", of the vertex corner. The latter situation results when the vertex is part of a hole in the face. In general, the side on which the web face lies is not known at creation time.

After all web faces have been created, those that represent corners of a single face are merged, as explained next.

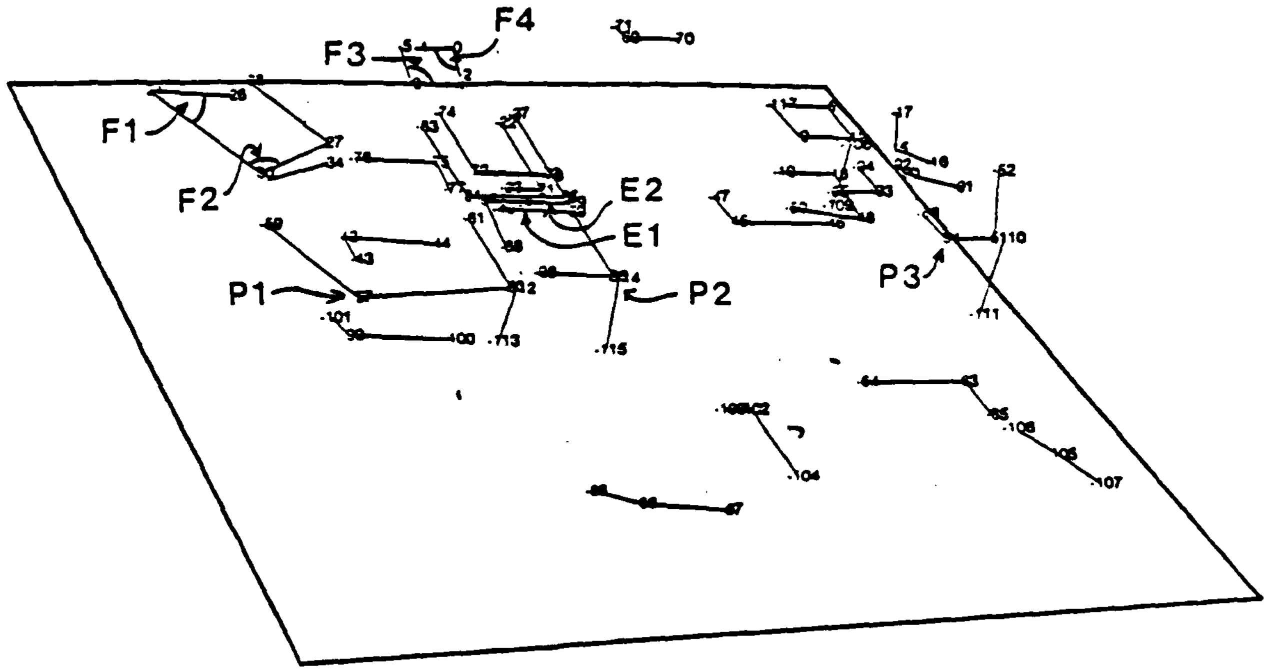


Figure 4-23:. Perspective view of 3D vertices and edges.

4.6.1.3 Merging Partial Faces

A face is partial if it is not complete, i.e., all of its edge-groups do not form closed edge rings. One way to complete a partial face is to merge it with nearby partial faces which represent different portions of the same face. The procedure that merges two nearby partial faces distinguishes two situations: (1) two faces that are touching, i.e., they share an edge (e.g., F1 and F2 in Fig. 4-23), or (2) two faces that are not touching (e.g., F3 and F4 in Fig 4-23).

Two partial faces that touch each other are merged if they satisfy the following conditions:

1. They must share exactly one edge (by definition of touching). Fig 4-26a depicts two partial faces, *f1* and *f2*, that share the edge *e2*.

2. The shared edge must serve as a boundary of both faces, but cannot partition them. This condition is satisfied if none of the vertices shared by the two faces lie on two edges of each face. In Fig. 4-26b, the partial faces *f1* and *f2* share the edge *e3*. These faces should not be merged because they share the vertex *v* which lies on two edges of each face (on *e2* and *e3* of *f1*, and on *e1* and *e3* of *f2*). Notice how *e3* serves to partition the faces, while in Fig. 4-26a, the edge *e2* serves as a boundary of the faces it joins.

3. The planes of the faces must be nearly parallel and very close to each other. This condition is tested by checking whether all the points lying on one face are within a threshold distance from

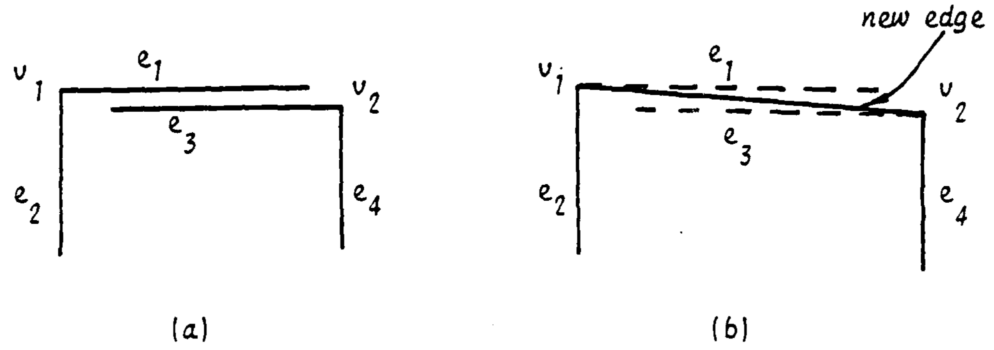


Figure 4-24: Combining edges. (a) Edges *e1* and *e3* are very close to each other, and each has a confirmed vertex. These vertices are on opposite ends of each other. (b) The new edge is shown as the result of merging *e1* and *e3*.

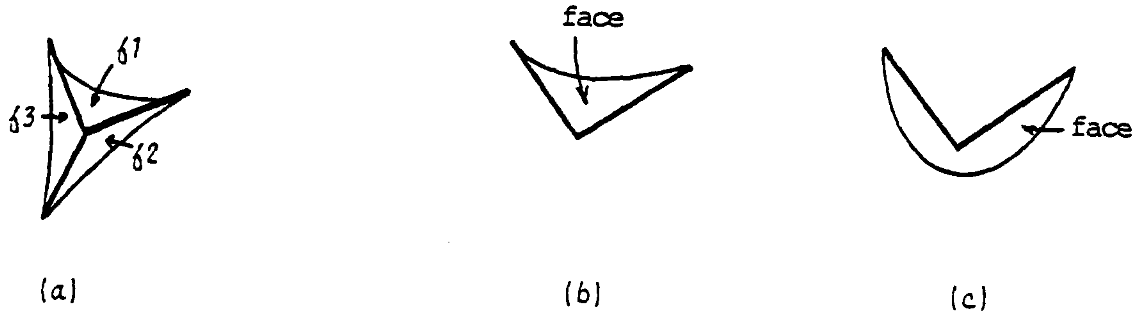


Figure 4-25: (a) Three web faces generated from a trihedral vertex. A web face may either be on the inside (b) or the outside (c) of a vertex corner.

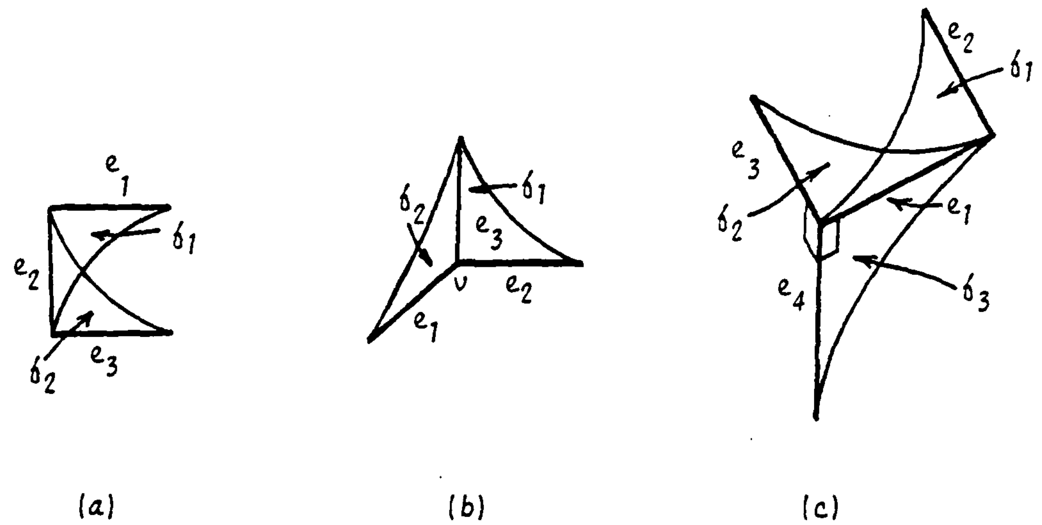


Figure 4-26: Situations for merging touching partial faces. (a) *f1* and *f2* share one edge. (b) *e3* partitions *f1* and *f2* rather than serving as a boundary for them. (c) *f1* and *f3* share an edge that bounds them, but they are not parallel.

the plane of the other face. In Fig. 4-26c, suppose *e4* is perpendicular to both *e1* and *e3*, and *e2* is parallel to *e3*. The partial faces *f1* and *f3* meet both conditions (1) and (2) above, but they do not meet the current condition.

The procedure for merging two touching faces F1 and F2 involves (1) finding the two edge-groups G1 of F1 and G2 of F2 that contain the shared edge, (2) subtracting edges and vertices from G1 (i.e., deleting part-of links in the structure graph) and adding them to G2, (3) subtracting edge-groups, edges, vertices, lines, and points from F1 and adding them to F2, and (4) recalculating the plane equation of F2 as a least squares fit to all the points now constraining F2.

Two partial faces that do not touch each other are merged if they satisfy the following conditions:

1. Each face must have an edge-group containing two non-vertex end points. In Fig. 4-27a, face *f1* has a single edge-group (consisting of edges *e1* and *e2*) that has the two non-vertex end points *p1* and *p2*. Similarly, face *f2* has a single edge-group with the non-vertex end points *p3* and *p4*.

2. Each of the two end points of the edge-group of one face must be uniquely matched with those of the other face. That is, each end point must be a distance of less than a threshold from exactly one of the two end points of the other face. In Fig. 4-27a, *p1* and *p3* are uniquely matched because their distance is less than the threshold and the distance from *p1* to *p4* is greater than the threshold. Similarly, *p2* and *p4* are uniquely matched.

3. The planes of the two faces must be nearly parallel and within a small threshold distance of one another.

The procedure for merging two non-touching faces is similar to the one for merging touching faces, in that elements are subtracted from one face and added to the other face, and the plane equation of the resulting face is recalculated. An additional step, however, involves finding the point of intersection of each pair of edges on which the matching pairs of end points lie. The points are then converted into new hypothesized vertices on the edges. The result of merging *f1* and *f2* in Fig. 4-27a is shown in (b), where two new vertices, *v1* and *v2*, have been hypothesized. Notice that the edge *e1* has been shortened in the process, while the other edges have been extended.

Up till now, we have only discussed the merging of partial faces. However, if the confirmed parts of two faces, each of which may be partial or complete, satisfy the three conditions outlined above for merging non-touching faces, then the faces may be merged. For example, suppose the face *f1* in Fig. 4-27c contains the confirmed edges *e1* and *e2* and the hypothesized edges *e3* and *e4*. Now suppose that the web face *f2* in (d) is new information that becomes available, say, from a new view. The confirmed parts of *f1* may then be merged with *f2* if they satisfy the conditions for merging. In the process, hypothesized parts of *f1* must be deleted. The mechanisms for doing this will be discussed later.

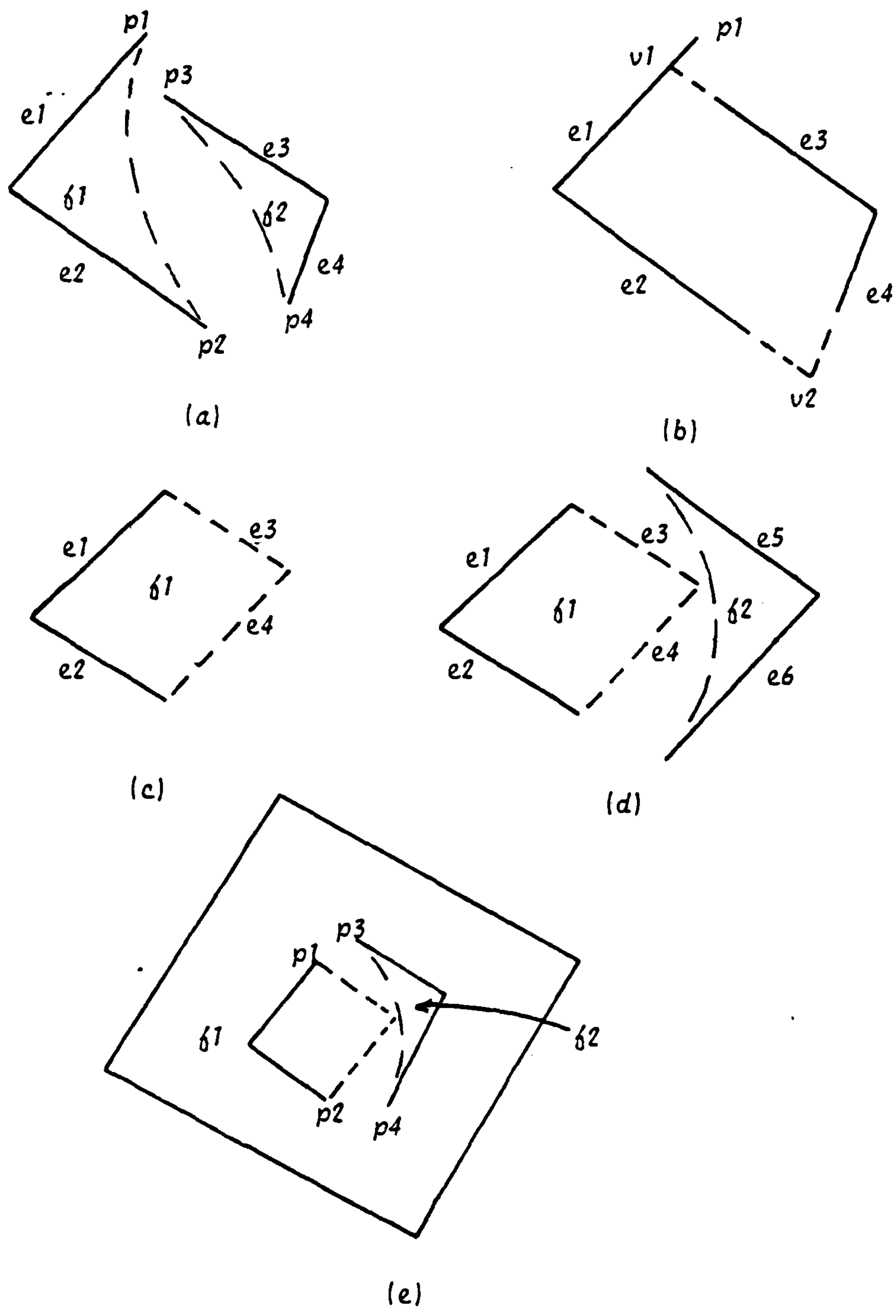


Figure 4-27: Merging of non-touching faces. (a) *f1* and *f2* satisfy the conditions for merging. (b) Result of merging *f1* and *f2*. (c) and (d) The complete face *f1* is merged with the partial face *f2*. (e) The complete face *f1*, which contains a hole, is merged with the partial face *f2*.

Another interesting example is depicted in Fig. 4-27e, whose situation is similar to that in (d) except that the web face *f2* is merged with confirmed parts of the face *f1*, which has a hole in it. Notice that the condition that the confirmed parts of each face must have two end points which are uniquely matched to those of the other face is satisfied by *p1* and *p3*, and by *p2* and *p4*. As a result of merging, *f2* aids in completing the boundary of the hole in *f1*.

After all mergers have been performed, many faces may still be incomplete. As will be explained later, knowledge of urban scenes is used to hypothesize the shapes of such faces, and they are completed by generating the appropriate edges and vertices.

4.6.1.4 Finding and Constructing Holes in Faces

The procedure for finding and constructing holes in faces occurs after all faces have been completed. The face F1 is assumed to represent a hole in the face F2 if the following conditions are satisfied:

1. The planes of the two faces are nearly parallel and within a small threshold distance of one another.

2. The bounding ring of vertices of F1, when projected onto the plane of F2, falls inside the boundary of F2.

If these conditions are satisfied, the edge-group that contains the bounding edges of F1 is subtracted from F1 and added to F2. It now serves as an inner edge-group (an inner ring of edges) of F2. F1 is then deleted from the structure graph.

4.6.2 Knowledge of Urban Scenes

Because the wire-frame data extracted from images represent a partial and sparse description of the scene, knowledge of planar-faced objects by itself is generally not adequate for completing many of the objects in the model. As will be described next, knowledge of urban scenes that contain block-shaped objects has been useful for this task.

4.6.2.1 Completing Shapes of Faces

Faces in the model may be incomplete because they contain one or more incomplete edge-groups, i.e., edge-groups without closed rings of edges. In these cases, the shape of each incomplete edge-group is hypothesized, and it is completed by generating the appropriate edges and vertices. The following rules are used here:

1. If the partial edge-group represents a single corner, i.e., it contains only two connected edges (the solid lines of face *f* in Fig. 4-28a), the shape is completed as a parallelogram. Two new edges are hypothesized to complete the shape and are added to the edge-group (dashed lines in the figure).

2. If the partial edge-group consists of three or more edges connected as a single chain (the solid lines of face *f* in Fig. 4-28b), the shape is completed by connecting the two end points of the chain with a new, hypothesized edge (dashed line in the figure), and adding it to the edge-group.

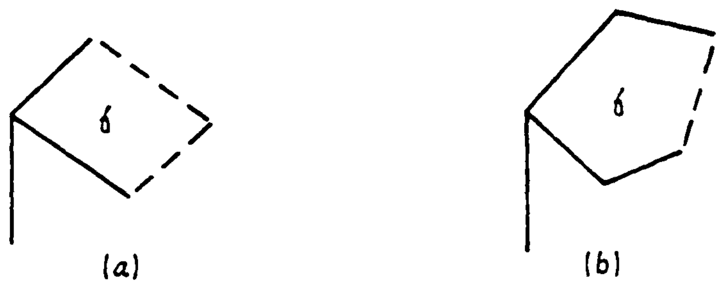


Figure 4-28: Completing shapes of faces. (a) The face *f* is completed in the shape of a parallelogram. (b) The face *f* is completed by closing the shape.

4.6.2.2 Hypothesizing Vertical Faces for Incomplete Objects

Objects in the model may be incomplete because they do not consist of a completely closed, connected set of faces. Since we are dealing with urban scenes, faces that lie high enough above the ground plane are assumed to represent roofs of buildings. A hypothesized vertical wall is dropped toward the ground from each edge of such faces, unless the edge is already part of another face.

The test to determine whether the face is high enough above the ground involves checking whether all the points on the face exceed a threshold distance from the ground. This test rules out faces that intersect the ground (such as building walls) or faces that lie on the ground (such as ground patches).

A vertical wall is dropped either to the ground plane or to the first face it intersects on the way down. For example, in Fig. 4-29a, face *f2* is above *f1*, and the distance of each from the ground plane exceeds the threshold. The result after dropping vertical faces is shown in (b), which indicates that faces have been dropped from *f1* to the ground, and from *f2* to *f1*.

The procedure for dropping vertical faces from a face F is as follows. For each vertex of F that has fewer than three legs, an edge is dropped either to the ground plane or to the first face it intersects. This results in a *vertical edge-frame* that supports F (the dotted lines in Fig. 4-29c). The edge-frame is then "filled in" by first creating web faces for each new edge pair at each vertex of F, then merging those that touch each other, and finally completing the resulting partial faces in the ways described earlier.

When the techniques described above are applied to the wire frames in Fig. 4-23, we obtain the scene model shown in Fig. 4-30. The planar patches at the "front" of the scene are part of the ground. Because they

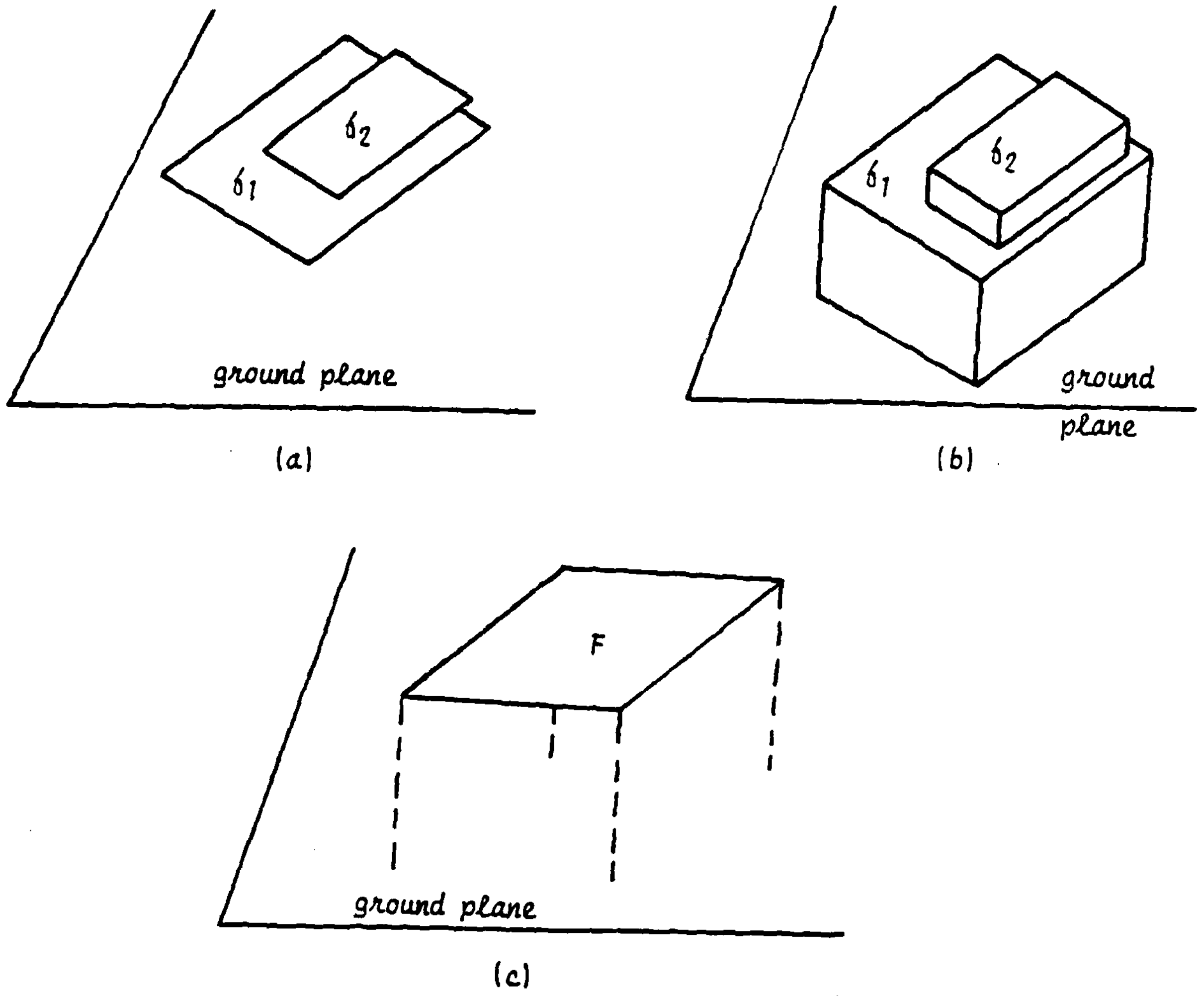


Figure 4-29: Dropping vertical walls from faces. (a) The face *f2* is above *f1*. (b) Faces are dropped from *f1* to the ground plane, and from *f2* to *f1*. (c) A vertical edge-frame is dropped from the face F.

were not high enough above the ground plane, they were not treated as building roofs. When these techniques are applied to the wire frames in Fig. 4-31, we obtain the scene model shown in Fig. 4-32. Note that all vertices, edges, and faces which have been hypothesized by the procedures described above are marked as such, and will be replaced by more correct versions as more information becomes available from new views.

4.7 Combining New Views with Current Model

The process of incorporating a 3D wire-frame description extracted from a new view into the current scene model can be divided into three main steps:

1. The wire-frame data must first be matched to the current model. This process provides (a) the

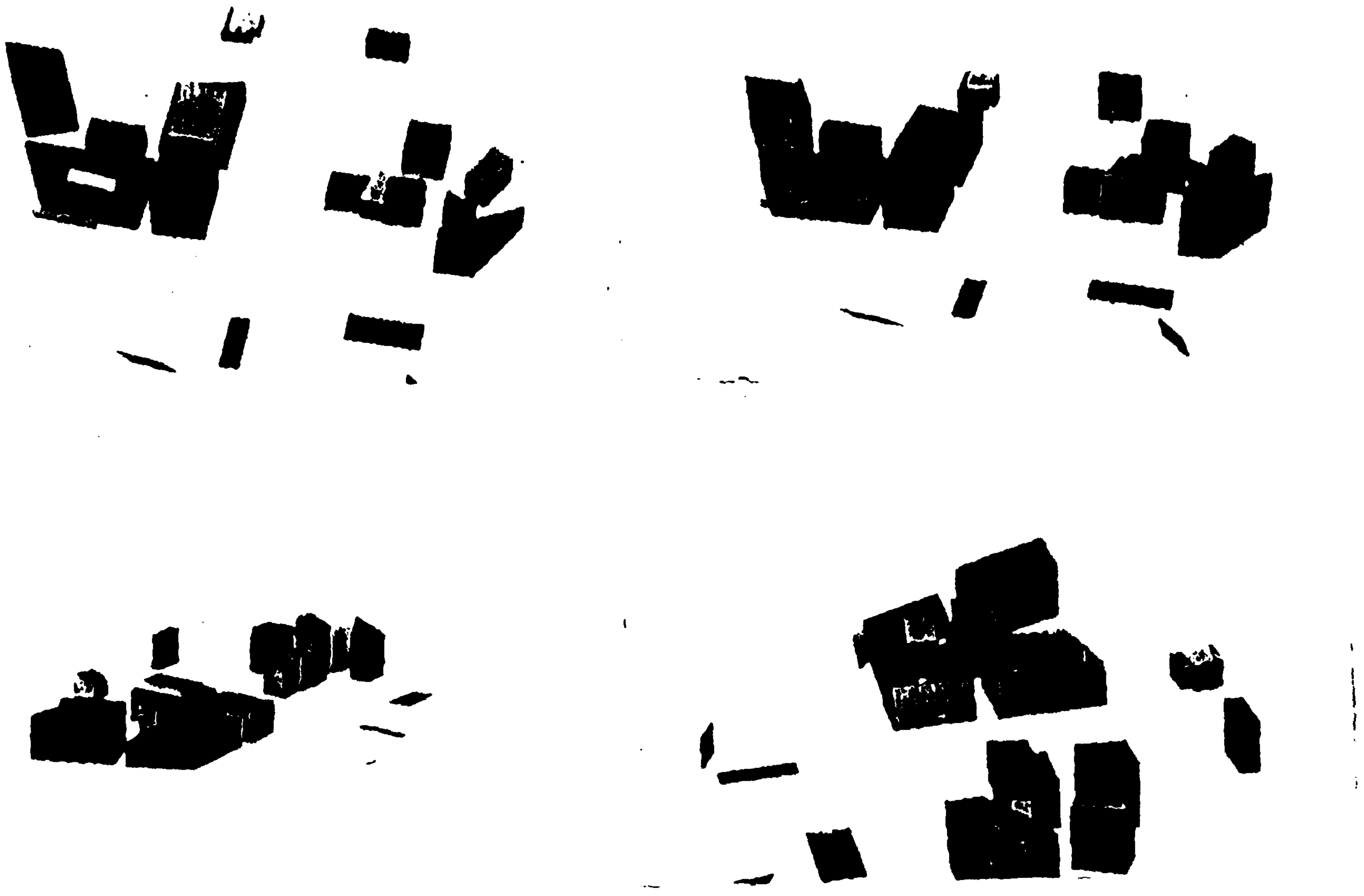


Figure 4-30: Perspective views of buildings reconstructed from wire-frame data in Fig. 4-23.

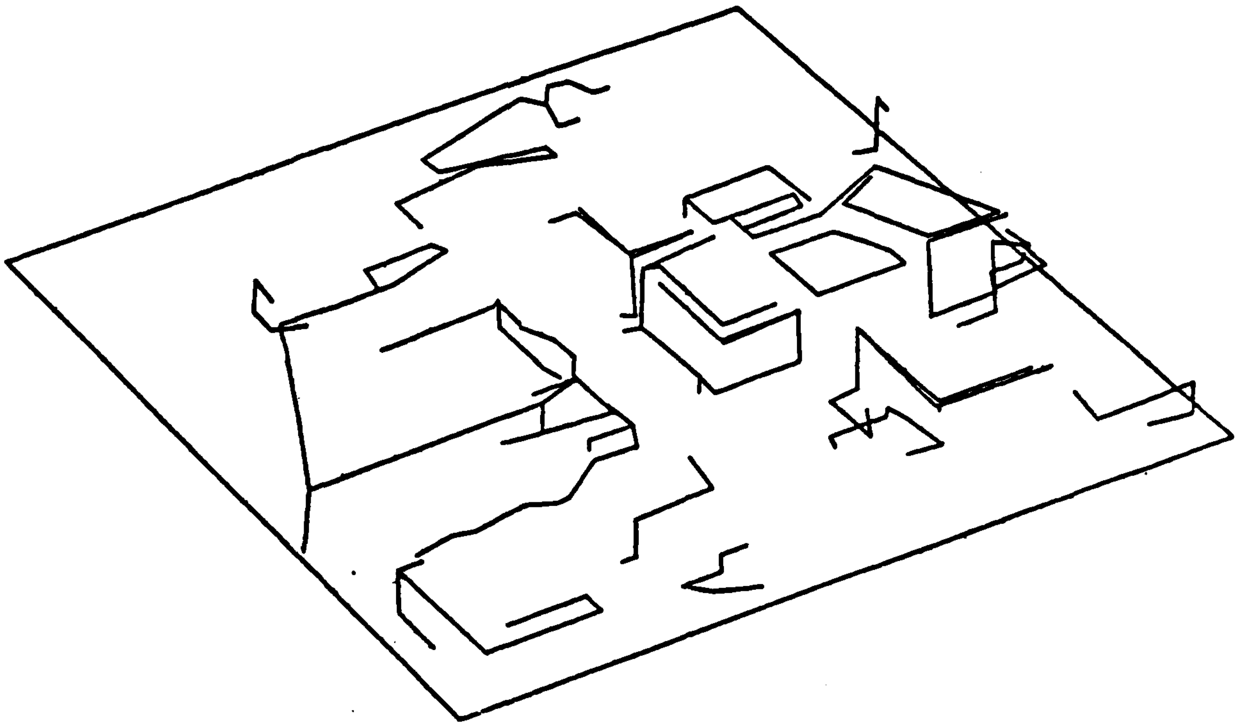


Figure 4-31: Perspective view of 3D vertices and edges.

scale transformation and coordinate transformation from the wire-frame data to the model, and (b) corresponding elements (i.e., vertices and edges) in the two.

2. The new wire-frame data is then merged with the current model. This process includes (a) merging pairs of corresponding elements, and (b) adding to the model wire-frame elements for which no correspondences were found. The latter procedure is aided by knowledge of the scale and coordinate transformations. During the merging process, hypothesized parts of the model that are inconsistent with the new wire-frame data are deleted.

3. At this point, many objects in the model may be incomplete because (a) new wire-frame data has been added, and/or (b) some hypothesized elements have been deleted. These objects are completed using the techniques described in the previous sections.

To see how these steps are carried out, consider the example of incorporating the information from a second view into the scene model of Fig. 4-30. This scene model was constructed from the set of wire frames (Fig. 4-23) automatically extracted from a "front" view of the scene. The second set of wire frames, shown in Fig. 4-33, was manually generated to simulate information available from an opposing point of view (viewing the scene from the "back"). The viewpoint for the perspective drawing of Fig. 4-33 is chosen to be similar to that of Fig. 4-23 to allow easier comparison by the reader. Notice that the information in Fig. 4-23 emphasizes edges and vertices facing the front of the scene, while those facing the back of the scene are emphasized in Fig. 4-33.

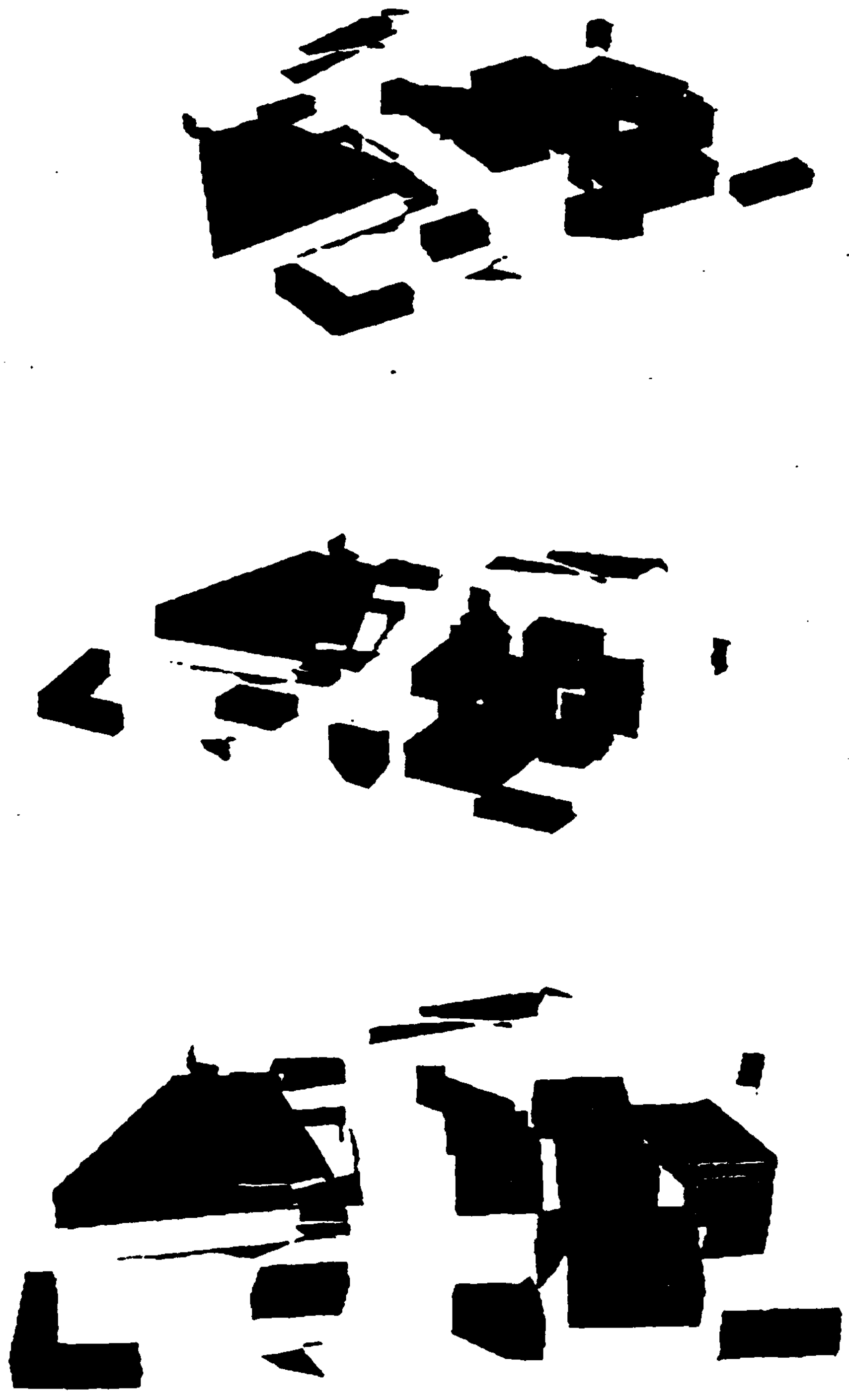


Figure 4-32: Perspective views of buildings reconstructed from wire-frame data in Fig. 4-31.

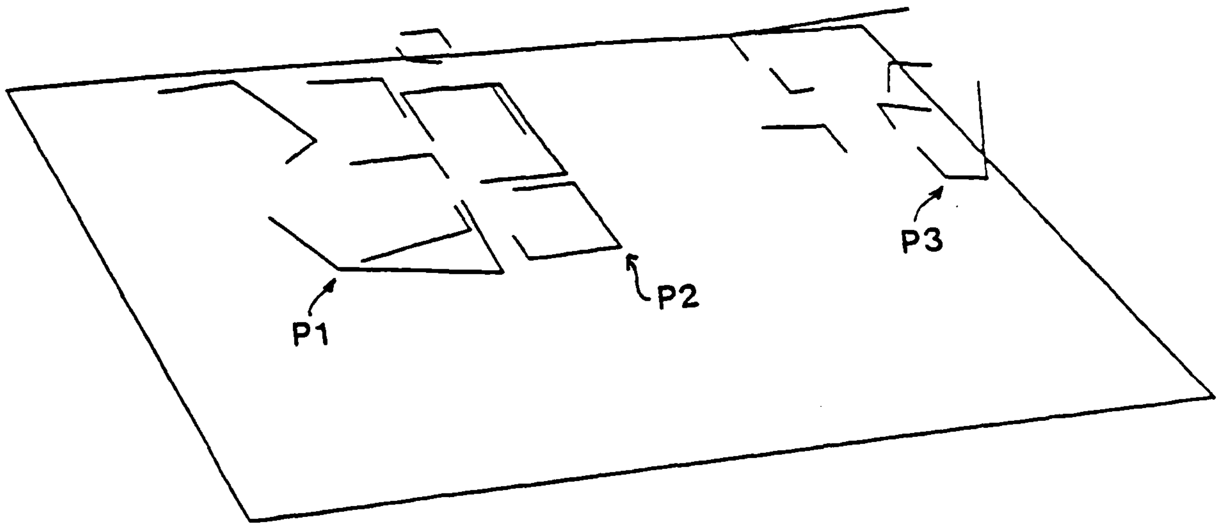


Figure 4-33: Perspective view of manually generated vertices and edges. The viewpoint for this drawing is chosen to be similar to Fig. 4-23. Points P1, P2, and P3, for example, correspond to points P1, P2, and P3 in Fig. 4-23.

4.7.1 Matching

We assume in this example that the scale and coordinate transformations from the new wire-frame data to the current model is known; the data and model may therefore be described in the same coordinate system. We have not yet implemented a general matcher that provides these transformations between the two.

The next step is to determine corresponding edges and vertices in the data and model. First we label each connected group of edges in the wire-frame data as a distinct wire-frame object. Next, wire-frame objects are matched with model objects. Two objects are said to match if they have confirmed parts that match. Matches are sought only for edges and vertices, since these constitute the only confirmed parts of a wire-frame object. The requirements for two confirmed vertices, one from each object, to match are: (1) they must be very close to each other, or (2) they must be part of matching edges whose other two vertices match. The requirements for two confirmed edges, one from each object, to match are: (1) the two confirmed vertices of one edge must match the two of the other, or (2) one confirmed vertex on one edge matches one on the other, and the two edges are close together and overlap in their lengths. These rules are used in a relaxation algorithm to obtain matching vertices and edges.

As an example, consider Fig. 4-35. Suppose the object in (a) is part of the model. The edges represented by the solid lines *e1*, *e2*, *e3*, *e4* and *e12*, are confirmed. The edges represented by the dashed lines are hypothesized. Vertices *v1*, *v2* and *v3* are confirmed, while the others are hypothesized. (Note that there are also edges and vertices in this object hidden from our viewpoint; these will not be considered here.) Suppose the wire-frame object in Fig. 4-35b has been derived from a new view, and it has been transformed to register with the model object. The following algorithm is used to match the two.

1. Find pairs of confirmed vertices that match by determining which ones lie within a threshold distance of one another. The vertices *v2* and *v100* are found to match, but let us suppose the distance between *v3* and *v101* exceeds the threshold.

2. Find pairs of confirmed matching edges that contain previously found matching vertices. The edges *e2, e3* and *e100, e101* contain matching vertices and are therefore compared. In order to match, two edges must be very close together and must overlap in their lengths. The distance threshold for this test, however, is greater than the one for determining matching vertices. This is permitted because the possible matching edges are also constrained by the requirement that they contain matching vertices. Therefore, even though *v3* and *v101* failed to match in step (1) above, the edges *e3* and *e101* are found to match, as are *e2* and *e100*.

3. For each new matching pair of edges found, if they contain a single pair of matching vertices, match their other vertices (if they exist and are confirmed). The vertices *v3* and *v101* match because *e3* and *e101* match. No new matching vertices result from the matching edges *e2* and *e100*, since *e100* has only one vertex.

4. Proceed by repeating step (2) above, i.e., find new pairs of confirmed matching edges that contain previously found matching vertices. The edges *e4* and *e12* are compared with *e102* and *e104*. Using the distance and overlap tests, *e4* and *e102*, as well as *e12* and *e104*, are found to match.

5. Next, step (3) is repeated. New matching vertices are sought that lie on newly found matching pairs of edges. The matching edges found in step (4) contain no new matching vertices, since *v4* and *v6* are unconfirmed. The algorithm therefore halts at this point; it would have continued with step (2) if new matching vertices had been found. The following pairs of matches are returned: *(v2, v100), (v3, v101), (e2, e100), (e3, e101), (e4, e102), (e12, e104).*

4.7.2 Discrepancies

We must now merge the new wire-frame data into the model. An important issue here is how to handle discrepancies between the two. We consider the following two types of discrepancies:

1. After the coordinate system of the wire-frame data has been transformed to that of the model and scale adjustments have been made, corresponding pairs of confirmed vertices and edges may not register perfectly in 3-space. In order to merge them into single elements, we perform a "weighted averaging" of their positions.

2. Hypothesized elements in the model may be inconsistent with newly obtained elements. We handle this by deleting such hypothesized elements.

To determine whether or not hypotheses are still valid when confirmed elements in the model are modified or deleted, we consider the elements which gave rise to the hypotheses. A hypothesis is dependent on all elements whose existence directly resulted in the creation of the hypothesis. If one of these elements is modified or deleted, the hypothesis must also be modified or deleted since the conditions under which it was created are no longer valid. The dependency relationships for hypothesized elements are explicitly recorded at the time of their creation using dependency pointers [Doyle 81].

We currently record these relationships for the following situations:

1. When two non-touching partial faces are merged (Fig. 4-34a), each face has two partial edges which are intersected with their counterparts in the other face. The intersection points form two new hypothesized vertices, each of which is dependent on the two edges whose intersection gave rise to it. In Fig. 4-34a, the arrows indicate the dependencies. Vertex *v1* is dependent on edges *e1* and *e3*, and vertex *v2* is dependent on edges *e2* and *e4*. If one of the edges were to be modified (e.g., if its position were to be displaced), the vertex that depends on that edge would no longer be a valid hypothesis, and would therefore be deleted. A new vertex might then be hypothesized.

2. When an incomplete edge-group is completed in the shape of a parallelogram (Fig. 4-34b), two new edges and three new vertices are hypothesized. Each of the new edges *e3* and *e4* is dependent on both of the old edges *e1* and *e2*. The edge *e3*, for example, is dependent on *e1* in the sense that its end point is constrained by the end point of *e1*. It is dependent on *e2* in the sense that it is constrained to be parallel to *e2*. The new vertex *v3* is dependent on the two hypothesized edges *e3* and *e4*, while the new vertices *v1* and *v2* are dependent on the confirmed edges on which they lie.

3. When a face is completed by connecting its two end points (Fig. 4-34c), two new vertices and one new edge are hypothesized. The new edge *e4* is dependent on both *e1* and *e3*, while the new vertices *v1* and *v2* are dependent on the edges on which they lie.

4. When a vertical wall is dropped from a face, the first step is to drop hypothesized edges from vertices of the face. Such edges are dependent on the vertices from which they are dropped. In Fig. 4-34d, the new edges *e1* and *e2* are dropped from, and are dependent on, the vertices *v1* and *v2*, respectively. A dropped edge is constrained to be perpendicular to the ground plane, and would therefore no longer be a valid hypothesis if the vertex it depends on, which is one of its end points, were to be displaced. After edges are dropped from all vertices of the face, the resulting edge-frame is filled in with faces, as described previously. This results in more hypothesized edges and vertices. The situations under which these are created fall under categories (2) and (3) above.

When a confirmed edge or vertex in the model is modified or deleted, the set of all hypothesized elements that depend on it are deleted. Recursively, elements depending on deleted ones are also deleted. When hypothesized vertices and edges are deleted in this manner, it is possible for hypothesized faces to lose minimal support, i.e., they may no longer be constrained by at least three non-colinear points. Such faces are also deleted.

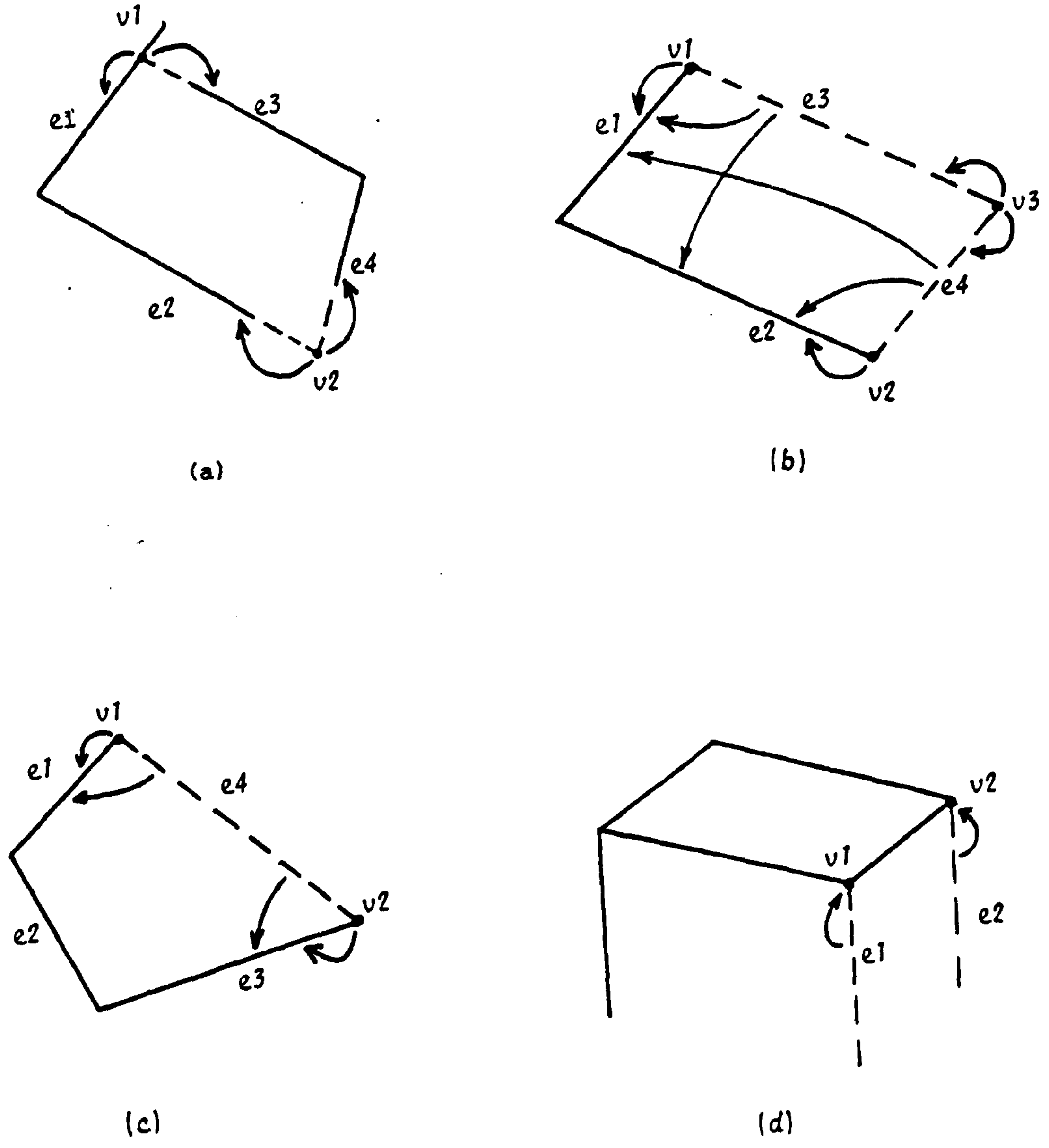
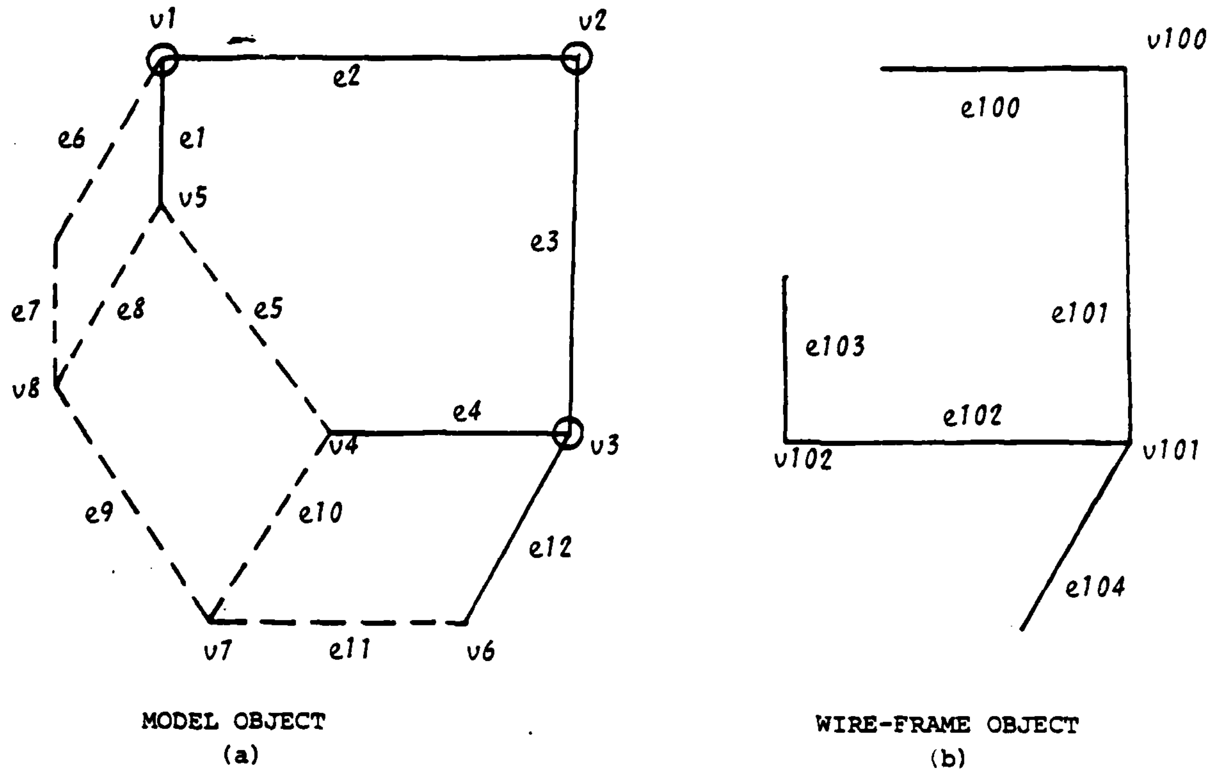


Figure 4-34: Generating dependencies for hypothesized edges and vertices. The dependence of an element on another is depicted as an arrow from the former to the latter. (a) Two non-touching partial faces are merged. (b) A face is completed in the shape of a parallelogram. (c) A face is completed by connecting its two end points. (d) Vertical edges are dropped from a floating face.



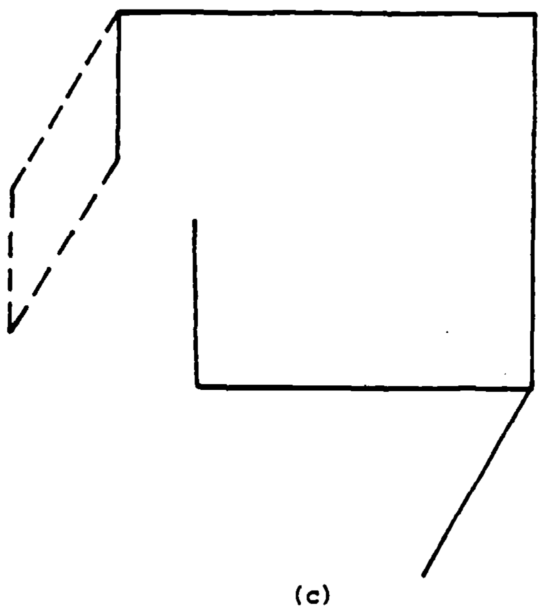


Figure 4-35: The wire-frame object in (b) is to be merged with the model object in (a). The confirmed edges of the model object (indicated by solid lines) are *e1*, *e2*, *e3*, *e4*, and *e12*; the confirmed vertices (indicated by circles) are *v1*, *v2*, and *v3*. Dashed lines represent hypothesized edges. (c) The result after merging.

4.7.3 Merging

The procedure that merges corresponding wire-frame and model objects takes into account the fact that the 3-space positions of end points of edges that are confirmed vertices are generally much more accurate than the positions of non-vertex end points. Therefore, confirmed vertices are given more weight during merging. As an example, consider again Fig. 4-35, where the wire-frame object in (b) is to be merged with the model object in (a).

The merging procedure starts by merging corresponding vertices in the two objects. This involves the following:

1. The model vertex of each corresponding pair of vertices (*(v2, v100)* and *(v3, v101)* in Fig. 4-35) is assigned new coordinates -- those of the midpoint of the line connecting the two initial vertices.

2. If the distance between the initial and resulting points of the model vertex exceeds a threshold, all hypothesized edges and vertices in the model that recursively depend on this vertex are deleted. Hypothesized faces that have lost minimal support are also deleted.

3. The vertex in the wire-frame object is deleted and replaced by the model vertex.

At this point, all corresponding pairs of edges will share at least one vertex. The corresponding edges are merged next as follows:

1. If the two edges share both their vertices (Fig. 4-36a), merging involves recalculating the line equation of the model edge and deleting the wire-frame edge. In Fig. 4-35 this situation occurs between edges *e3* and *e101*.

2. If the two edges share one vertex but only one of them contains another confirmed vertex (Fig. 4-36b), the edge with one confirmed vertex is deleted, leaving the edge with two vertices as the result. In Fig. 4-36b, the result of merging *e1* and *e2* is *e1*. Notice that the non-vertex end point in this case is given zero weight. If the resulting edge is from the wire-frame object, it is subtracted from this object and added to the model object. In Fig. 4-35, this situation occurs between edges *e2* and *e100*, and edges *e4* and *e102*.

3. If the two edges share one vertex and the other end points are not confirmed vertices (Fig. 4-36c), the line equation of the new edge is obtained by a least squares fit to all the points on the two initial edges (see Fig. 4-36d, where the dotted lines are the initial edges, and the solid line is the new edge). The non-vertex end point of the new edge is the projection of the non-vertex end point of the longest initial edge onto the line constraining the new edge. This new end point is labeled as confirmed. The edge is then added to the model object and the two initial edges are deleted. Note that the vertex end point of this edge need not lie on the line constraining the edge. In Fig. 4-35, this situation occurs between edges *e12* and *e104*.

Before merging, a model edge may contain either one confirmed vertex or two confirmed vertices. If it contains one confirmed vertex, then all hypothesized edges and vertices in the model that recursively depend on this edge are deleted. Hypothesized faces that have lost minimal support are also deleted. In Fig. 4-35,

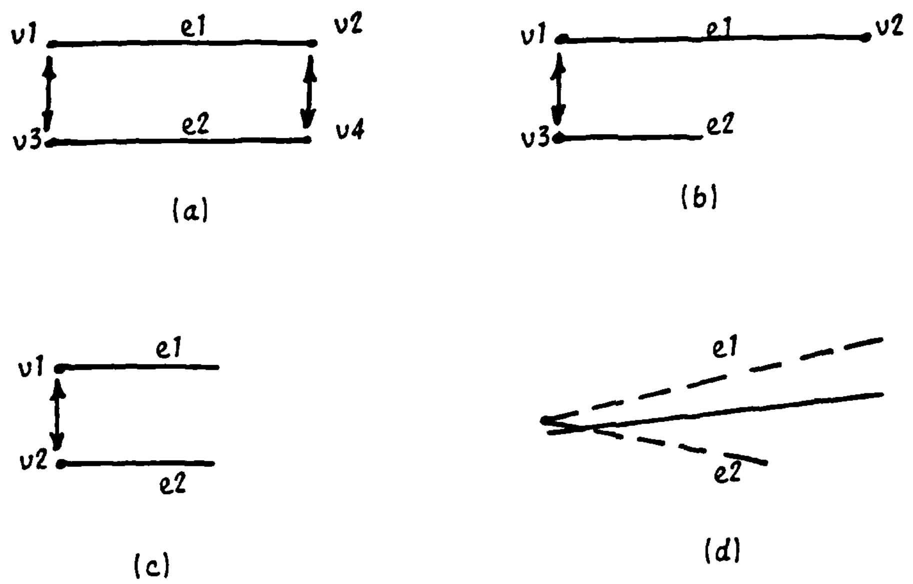


Figure 4-36: Merging edges. Two edges to be merged may either (a) share both their vertices, or (b) and (c) share one vertex. (d) Result of merging edges in situation (c).

this occurs for the edges *e4* and *e12*. The hypothesized elements in the figure that recursively depend on, say, *e4* are the vertices *v4* and *v7*, and the edges *e5*, *e10*, *e9* and *e11*. If a model edge to be merged contains two confirmed vertices (e.g., *e2* and *e3* in Fig. 4-35), no hypothesized elements need be deleted since all necessary deletions were made when the vertices of the edge were merged.

After all corresponding elements of the two objects have been merged, the edges and vertices remaining in the wire-frame object that were not merged are added to the model object, and the wire-frame object is deleted. In Fig. 4-35, this step involves adding the edge *e103* to the model.

Finally, the plane equation is recalculated for each face in the model object which had edges and vertices that were modified or deleted. Fig. 4-35c shows the final configuration of the object after the merging process. This object is incomplete and must be completed using the techniques described in previous sections.

4.7.4 Results of Merging

When these procedures are applied to the wire-frame data in Fig. 4-33 and the scene model in Fig. 4-30, we obtain the updated scene model shown in Fig. 4-37. The updated version has two important improvements over the initial version. First, the updated model contains more buildings since new wire-frame data, some of which represent new buildings, have been incorporated into the initial model. Second, for many buildings

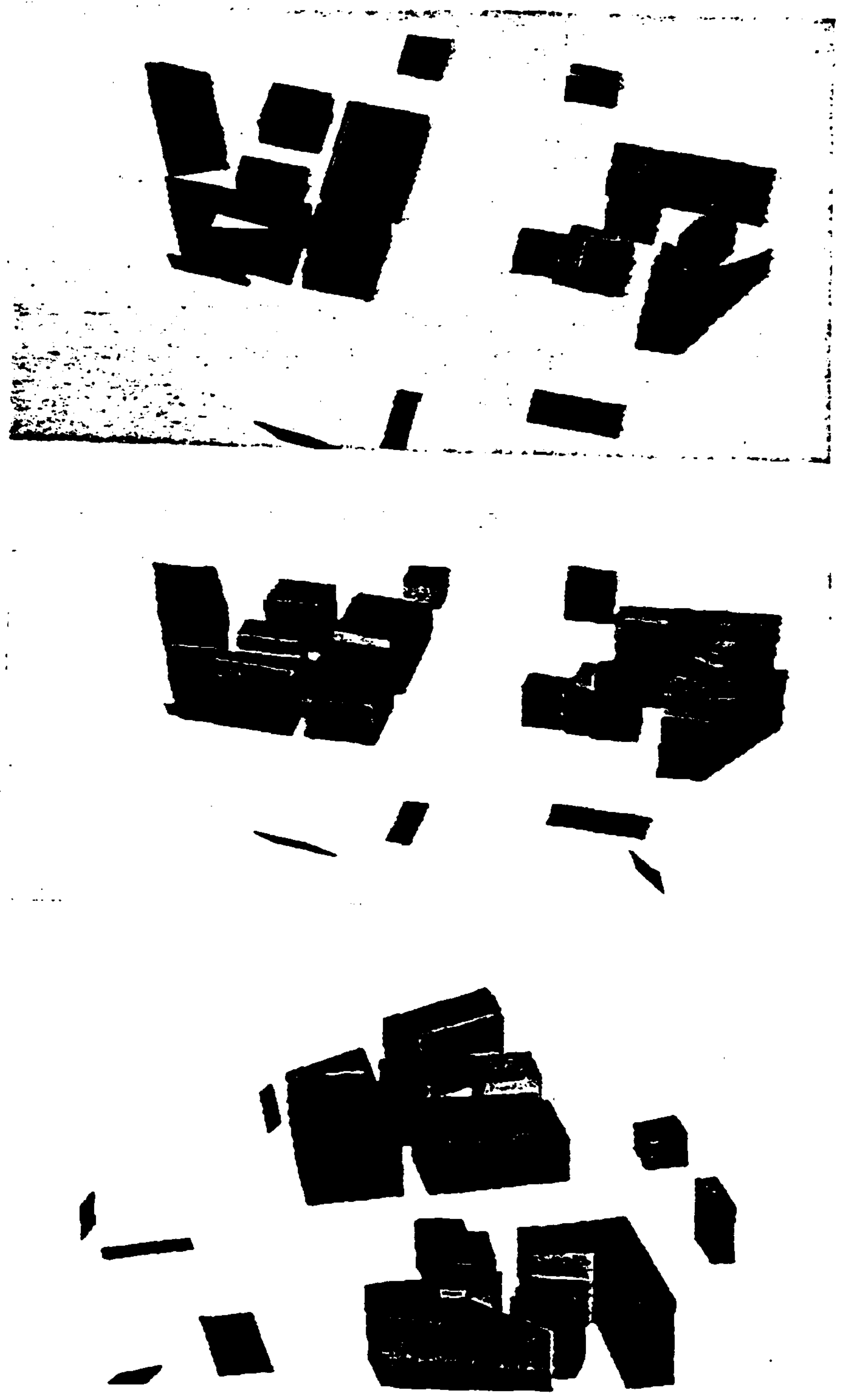


Figure 4-37: Perspective views of buildings derived by incorporating the wire-frame data in Fig. 4-33 into the model in Fig. 4-30.

described in both versions of the model, the positions of vertices and edges are more accurate in the updated version. This is because many hypothesized vertices and edges are replaced by accurate ones obtained from the new data, and many confirmed vertices and edges are merged with corresponding ones in the data by "averaging" their positions, generally decreasing the amount of error.

This experiment demonstrates how information provided by each additional view allows the model to be incrementally made more complete and accurate.

4.8 Summary

In this chapter, we have concentrated mainly on the problems of extracting information from high resolution aerial images of urban scenes, and accumulating the information in a 3D scene model. We have presented results in two aspects of the 3D change detection task: the low-level problem of analyzing images, and the high-level problem of representing, constructing, and updating the 3D scene model.

In the low-level processing, we have described techniques for extracting building structures from the images, and we have described experiments which determine how to efficiently search a line image in order to form junctions. In the high-level processing, we have described how the scene model, a surface-based description, is represented and incrementally acquired from a sequence of images.

References

[Baer, Eastman, and Henrion 79]
Baer, A., Eastman, C., and Henrion, M.
Geometric Modelling: a Survey.
Computer-Aided Design 11:253-272, September, 1979.

[Doyle 79] Doyle, J.
A Truth Maintenance System.
Artificial Intelligence 12 :231-272, 1979.

[Doyle 81] Doyle, J.
Three Short Essays on Decisions, Reasons, and Logics.
Technical Report STAN-CS-81-864, Department of Computer Science, Stanford University, Stanford, CA, May, 1981.

[Duda and Hart 73]
Duda, R.O. and Hart, P. E.
Pattern Classification and Scene Analysis.
John Wiley and Sons, New York, 1973.

[Eastman and Preiss 82]
Eastman, C. M., and Preiss, K.
A Unified View of Shape Representation and Geometric Modeling.
Israel Conference on CAD , January, 1982.

[Nevatia and Babu 80]
Nevatia, R. and Babu, K.R.
Linear Feature Extraction and Description.
Computer Graphics and Image Processing 13:257-269, July, 1980.

[Requicha 80] Requicha, A. A. G.
Representations for Rigid Solids: Theory, Methods, and Systems.
Computing Surveys 12(4):437-464, December, 1980.

5. SUMMARY

5.1 SUMMARY

This phase of the project focused on the development and evaluation of methods which yield representations of structural and textural information in an image, and relate these representations to object and surface contour properties of the scene. These representations will be used as the basis for modeling and interpolation of time-varying properties of scenes. The techniques currently being studied include (Chapter 2):

- *Probabilistic Graph Matching* - Attributed graph structures are used as the basis for a composite structural-statistical model of information in an image. Structural elements such as edges, corners, regions, or local basis function expansion peaks are used as elements of the graph structure. Matching of object classes, of stereo pairs, and of sequential time-varying scenes have been studied.

- *Multiple Resolution Structural Basis Functions* - Local expansion of the gray level intensities of an image is carried out at multiple resolution levels. This hierarchical representation of the image is useful for the detection and recognition of objects, and facilitates the description of tracking of time-varying objects at low resolution with updating of object description at higher levels.

- *Textural Surface Models* - The relationship between image texture and surface contours is fundamental to the interpretation of images with textured regions and particularly to the interpretation of variations in textured regions over time. These studies are looking at the relationship between two-dimensional random field models of surface contour and observed intensity fields under known conditions of illumination and imaging.

The algorithms described in this section reflect the interdisciplinary nature of the overall project. The structural basis function and texture models described above are particularly well-suited to parallel or optical processor implementation. They will be implemented and evaluated on the array processor with RAPIDbus host described in Appendix A. This architecture will provide the basis for integration of parallel preprocessing algorithms with representations which are well-suited to model-based interpretations. The interactive use of parallel and optical preprocessing with hypothesis formation and adaptive search strategies is a longer term goal of this research.

Other algorithms for parallel feature extraction have been addressed and will be reported on in later reports. These include moments, chords, Fourier coefficients and synthetic discriminant functions.

All of these features are capable of being optically-generated. A considerable amount of effort has also been spent in assembling two digital processing facilities for use in this program. This hardware is detailed in Appendices A - B.

A major effort on the extraction of time-varying sub-pixel targets in noise has been achieved. This time-change scenario concerns applications such as the detection of missile launches or aircraft in flight. A staring mosaic sensor with frame rates of 0.01 seconds is assumed. In this first year, we have achieved the following results on this specific time-change scenario (Chapter 3).

- We have produced synthetic space-based imagery with correlated noise, uncorrelated noise, sub-pixel targets, separate background and target sub-pixel shifts, with controlled correlation noise statistics.
- We have evaluated three estimators, all of which were demonstrated to achieve excellent sub-pixel registration accuracy.
- We have evaluated the performance of three interpolators for use in registering sub-pixel shifted image frames. These tests showed much success with a considerable reduction in the mean square error after the image subtraction.
- We have evaluated the correlation plane dynamic range required and found it to be quite low since only several sampled points about the center of the correlation plane need be measured.
- We have evaluated the accuracy with which each separate correlation plane sample must be measured. We have found the system to provide excellent sub-pixel registration accuracy even in the presence of large uncorrelated noise present in the input imagery.
- We have successfully demonstrated the performance of the system and its ability to detect and track sub-pixel targets. This is the first time this has been achieved, to our knowledge.

The problem of detecting three-dimensional changes in a complex urban scene is a very difficult one, particularly since any information extracted from the complex images is highly incomplete and contains many errors. Therefore, we have thus far concentrated mainly on the problems of extracting information from such images and accumullating the information in a 3D scene model. Future work will include the problem of detecting and dealing with changes in the scene.

In this report, we have presented results in two aspects of the 3D change detection task: the low-level problem of analyzing images, and the high-level problem of representing, constructing, and updating the 3D scene model (Chapter 4).

In the low-level processing, we have described techniques for extracting building structures from high resolution aerial images of urban scenes. Edge points are first extracted from an image, and then straight line segments are fitted to them. Junctions are then formed from the line segments. These junctions are used to assign the segments to a structural model of buildings. A search using a Hough transform is then performed to look for new line segments predicted by the model.

A fundamental problem in interpreting complex images is to relate image features to scene features. In our context, this involves distinguishing two classes of image line segments, those arising from building boundaries and those arising from texture or shadow boundaries. We handle this problem by utilizing task specific knowledge. We assume that lines forming junctions arise from building corners only if one of the lines is vertical in the scene, i.e., is directed toward the vertical vanishing point. These lines are then labeled as part of a building model that consists of an arbitrary number of connected vertical faces covered by a roof. Lines that are not consistent with this building model are assumed to arise from texture or shadow boundaries.

In the low-level processing, we have also described experiments which determine how to efficiently search a line image in order to form junctions. Each line segment in the image is represented as a unique unit containing the x,y coordinates of the two end points. The set of line segments are stored as a list. A simple but inefficient way to determine the lines that lie within a small window in the image is to test each line in the list. The access time can be improved by dividing the image into a number of small areas called sectors. Each sector has a list of the lines in its area. The search now requires only that the lists of the sectors containing the window be searched. We have empirically determined that the fastest access time is obtained when the image is divided into sectored areas forming from 6 to 8 rows and columns.

In the high-level processing we have described, techniques for representing, constructing, and updating the scene model. The scene model is a surface-based description of an urban scene, and is incrementally acquired from a sequence of images. Each view of the scene undergoes analysis which results in a 3D wire-frame description that represents portions of edges and vertices of buildings. The initial model, constructed from the wire frames obtained from the first view, represents an initial approximation of the scene. As each successive view is processed, the model is incrementally updated and gradually becomes more accurate and complete. Task-specific knowledge is used to construct and update the model from the wire frames.

The model is represented as a graph in terms of symbolic primitives such as faces, edges, vertices, and their topology and geometry. This permits the representation of partially complete, planar-faced objects. Because incremental modifications to the model must be easy to perform, the model contains mechanisms to (1) add primitives in a manner such that constraints on geometry imposed by these additions are propagated throughout the model, and (2) modify and delete primitives if discrepancies arise between newly derived and current information. The model also contains mechanisms that permit the generation, addition, and deletion of hypotheses for parts of the scene for which there is little data.

5.2 FUTURE WORK

- *Probabilistic Graph Matching* - Further studies of attributed graph structures as a basis for image matching and scene recognition will focus on search strategies for matching subject to geometric constraints, optimal selection of image subpatterns for matching in noisy images, and incorporation of time-varying attributes to scenes with common structure. Using the preprocessing capabilities for attributed graph extraction developed during the previous year, we will apply the current techniques and investigate these new issues for aerial scenes.

- *Texture Analysis* - Work in texture shall concentrate on discriminating between different 2-dimensional and 3-dimensional textures and extracting information such as viewing angle, light incidence angle and gradient. A set of texture measures, each of which offers reliable, consistent discrimination between the various types will be chosen for this task. From the perspective of high-altitude imagery, the goal in mind is be able to distinguish between different types of vegetation (e.g. forests, crops, grasslands) and different types of terrain (e.g. desert, hills, mountains).

- *Structural Analysis* - Future work in structural analysis will explore model based structural image analysis using iteratively generated local shape experts. Specific subgoals include the development of appropriate object representations (hierarchically structured, planar models seem promising), algorithms for building a global scene model using iterative assignments and responses from local shape experts, and designs for adaptable shape experts. The initial evaluation object domain will center on planes, vehicles, and buildings found in airfield photography.

- *Supporting Architecture* - Continuing work on the RAPIDbus multiprocessor system will extend hardware and software capabilities so as to provide a unique support environment for future project research. An immediate system is being constructed with at least two array processors, eight general purpose processors, and high speed data transfer capabilities. Software development will provide an environment in which many of the algorithms discussed in chapter two can be implemented so as to make *effective use* of the RAPIDbus architecture.

In the area of optical processing and extensions of our Chapter 3 research, there are several areas of research that should be pursued to analyze high frame rate sub-pixel target time-change imagery. These include both algorithmic advances and simulation studies.

All sub-pixel shift estimators we considered so far have been ad hoc in nature and are non-parametric. We will extend them to parametric methods by assuming proper models for the underlying pdfs. This will lead to optimal sub-pixel shift estimators based on maximum likelihood (ML), maximum a *postesiori* (MAP) and minimum mean squared error (MMSE) strategies. We will also investigate the statistical behavior of these estimators to assess their true performance.

We have considered only 2 frames of our time-change imagery to suppress the background and extract the target. We will treat the target extraction problem in a general 3-D space (x,y coordinates of the image and time t) and investigate general 3-D signal processing methods to provide the target track. This will require radically different strategies for target extraction than the simple subtraction we are currently using.

Our gradient based estimator requires further study to determine the reason for its larger number of iterations. Our LMSE estimator also requires further analyses (with respect to the condition number of the matrix) as does our linear interpolator (specifically its convergence). Multi-level cloud imagery will be produced, 3-D aircraft imagery will be generated and inserted in this imagery and 3-D data distortions will be produced and tested (using multiple correlation plane central regions for registration).

In the area of optical feature extraction, the optical generation of various object features (moments, chords, etc.) and generic object information (structural and textural via generic SDFs) will be addressed. The AI and IU aspects of this work will be addressed in years 3 and 4 of this research effort.

There are many areas of research that we will pursue in the future for the 3D change detection task. These include the following:

1. Interactively generate an initial scene model. This will assure that the model is reasonably accurate and compiete. When 3D information from a subsequent view is compared with it, we can have more confidence that discrepancies between the new information and the model are due to changes in the scene, rather than errors in analysis. Also, the model may be used to interpret the image in a top-down manner.

2. Apply stereo and monocular techniques for extracting 3D scene information.

3. Develop methods for matching the extracted scene information with the current model. Such a procedure is necessary to register the new information with the model so that they can be compared to determine changes in geometry and structure. One idea is to match vertices in the two descriptions, since each pair of matching vertices suggests a possible coordinate transformation.

4. Develop methods for interpreting discrepancies between newly extracted 3D information and the current model. Such discrepancies may be due either to the scene having changed or to errors in the new information and/or model. One way of interpreting discrepancies is to assign confidence values to the newly extracted information. Discrepancies involving highly confident information might imply that the scene has changes, while discrepancies involving information with low confidence might imply errors. A more powerful way of interpreting discrepancies is to use knowledge of the kinds of scene changes expected. This knowledge may either be known a priori, or may be induced from the pattern of changes that have previously occurred. In this way, discrepancies are interpreted as scene changes only if they are consistent with predicted scene changes.

5. Develop methods for generating up-to-date maps of the scene. The central, integrated scene model which is continuously updated with new information may also be used for generating up-to-date maps of the scene. Such maps require extracting global properties and features such as shape, size, orientation, and relative positions of objects in the scene. This kind of information is readily available from the scene model.

6. PUBLICATIONS, PRESENTATIONS AND STAFF SUPPORTED

6.1 STAFF SUPPORTED

Electrical and Computer Engineering --

D. Casasent (Professor), Principal Investigator

B.V.K. Vijaya Kumar (Assistant Professor)

Yeou-Lin Lin (Research Associate)

G. Kashipati (Research Assistant)

The Robotics Institute --

Arthur C. Sanderson (Professor), Principal Investigator

John Willis (Research Assistant)

Nanda Alapati (Research Assistant)

James McQuade (Research Programmer)

Computer Science --

Takeo Kanade (Professor), Principal Investigator

Martin Herman (Research Associate)

Duane Williams (Programmer)

6.2 PUBLICATIONS

Electrical and Computer Engineering --

1. B.V.K. Vijaya Kumar and C. Carroll, "Loss of Optimality in Cross-Correlators", JOSA-A, Vol. 1, 1984, pp. 392-397.

2. D. Casasent and V. Sharma, "Feature Extractors for Distortion-Invariant Robot Vision", Optical Engineering, November 1984 [accepted].

3. B.V.K. Vijaya Kumar, "Lower Bound for the Suboptimality of Cross-Correlators", Applied Optics, Vol. 23, July 1984 [accepted].

4. D. Casasent, A. Goutzoulis and B.V.K. Vijaya Kumar, "Time-Integrating Acousto-Optic Correlator: Error Source Modeling", Applied Optics [submitted].

The Robotics Institute --

1. J.L. Crowley and A.C. Sanderson, "Multiple Resolution Representation and Probabilistic Matching of 2-D Gray-Scale Shape", *Proceedings of the Workshop on Computer Vision: Representation and Control*, IEEE Computer Society, April 30-May 2, 1984, Anapolis, MD, pp. 95-105.

Computer Science --

1. None.

6.3 CONFERENCE PRESENTATIONS AND SEMINARS

Electrical and Computer Engineering --

1. D. Casasent, "Fourier Transform Feature-Space Studies", Presented at the SPIE Conference, November 1983, Cambridge, Massachusetts.

2. D. Casasent, "Synthetic Discriminant Functions", Presented at DARPA, February 1984.

3. D. Casasent, "Robotics Applications of Optical Data Processing", Presented at Polytechnic Institute of New York, February 1984.

4. D. Casasent, "Optical Pattern Recognition", Presented at the Air Force Office of Scientific Research, May 1984.

The Robotics Institute --

1. J.L. Crowley and A.C. Sanderson, "Multiple Resolution Representation and Probabilistic Matching of 2-D Gray-Scale Shape", presented at IEEE Computer Society Workshop on Computer Vision: Representation and Control, April 30- May 2, 1984, Anapolis, MD.

2. A.C. Sanderson, "Probabilistic Graph Matching Methods", presented at the International Conference on Computer Vision and Industrial Applications, May 14-18, 1984, Stockholm, Sweden.

Computer Science --

1. M. Herman, "Representation and Incremental Construction of a Three-Dimensional Scene Model", Presented at the Workshop on Sensors and Algorithms for 3-D Machine Perception, Washington, D.C., August 1983.

APPENDIX A
RAPIDBUS:
ARCHITECTURE AND IMPLEMENTATION

A.1 Architecture

RAPIDbus II is designed as a multiprocessor to provide system support for research into new approaches for the analysis of complex, space based imagery. The structure is based on the hypothesis that algorithms can be formulated as a multitude of largely concurrent, tightly coupled, computationally intensive tasks. A moderate size proof-of-concept machine is currently being fabricated with the assistance of several companies.

Broadly multitasking algorithms were explicitly chosen in an effort to escape the performance bounds of cost-effective uniprocessors. Although conventional single processor mainframes often support multiple communicating tasks, they do so by dividing the resources of one processor across all active tasks. Such a multitasking system provides no performance incentive to develop task level algorithmic concurrency. Executing on a multiprocessor support base, the researcher is potentially richly rewarded for the additional effort required to develop concurrent approaches. RAPIDbus II is directed at the challenge of minimizing the effort required to take advantage of task concurrency while providing application specific performance to support qualitatively new approaches to analysis systems.

A.1.1 Extensibility

A substantial overhead cost is paid at the systems software level when moving from one to two equal processors. Relatively few changes are needed to the software structure as additional processor are added to the initial pair. If a sufficient number of tasks are available, potential increases in system throughput are generally limited by the increasing average communications latency as tasks begin to block each other while accessing common data structures. Through implementation techniques evolved from earlier RAPIDbus designs, RAPIDbus II appears to be capable of minimizing both contention for interchange bandwidth, and with suitable hosts, contention for access to shared memory blocks [Zoccoli 81, Willis 82].

Since the number of tasks is an experimental variable, the ideal number of processors appears limited by the maximum acceptable average communications latency. A reasonable metric for such a radius of extensibility is based on the time required for a processor to respond to a context swap request and execute the desired task. If a process can get the results of a task execution faster

PREVIOUS PAGE IS BLANK

locally than by communicating with a distant processor, the foreign processor clearly lies beyond the range of practical extensibility.

Consideration of actual RAPIDbus II implementation technology suggests that a task swap has nearly 100 clock cycles of overhead before replacement code can begin execution. Several hundred RAPIDbus II processors could be placed within this communications latency. Thus in order to demonstrate reasonable extensibility without incurring substantial addressing overhead, 240 nodes[1] are defined by the architecture, with straight forward extension to a larger population.

Within a single task, address space limitations in previous multiprocessor systems have been recognized as a serious limitation to extensibility [Jones 80]. Processors such as the DEC PDP-11, and the Zilog Z80 used in several early multiprocessors are limited to a 2^{16} byte address space [Jones 80, Rieger 81]. Taking advantage of the tremendous leap in semiconductor technology, RAPIDbus II is designed around a 2^{32} byte physical address space. The virtual address space available to the programmer is host dependent, but in the proof-of-concept realization, provides up to 2^{24} bytes in one or more separately mapped segments[2].

A.1.2 Heterogeny of Elements

In traversing the depth of an advanced system from visual input, through image understanding, to a suitably enunciated output, a myriad of different kinds of algorithms are required. Early vision processing may require very regular operators on large data objects. Later stages of vision may require the manipulation of small objects interwoven within intricate data structures. Reporting or control stages will require still other computational support. Within a research environment, externally supplied input or data output may be required at any point in conjunction with a variety of devices.

Traditionally, multiprocessors have relied on a single general purpose execution element which is replicated as needed throughout the structure. Although such homogeneity simplifies the design, any general processor cannot hope to optimally handle a broad range of array, scalar, and I/O tasks in a cost- effective way. Trade-offs are inevitable with a general purpose architecture. In contrast, designers are increasingly finding ways to create processor structures which achieve very impressive performance over a limited algorithm domain [Kung 82]. For a given cost, performance often is inversely proportional to flexibility in the design of a processing element.

[1]The number of potential nodes includes both processor locations, and those used only for system communication [links].

[2]Bankswitching makes the remainder of the address space accessible with greater effort

A multiprocessor provides potential to take advantage of special purpose processor structures interfaced as one or more host nodes. If a complementary set of nodes is assembled so as to work symbiotically with one another, the performance of special purpose designs can be harnessed without loosing some generality of overall function. However, within a research environment, the cost of designing any processor module is high. Seldom do resources provide adequate support for both the hardware design and later software support of more than one processor architecture.

RAPIDbus thus relies heavily on being able to integrate the vast effort invested in existing commercial hosts, either at the chip set or subsystem level. Typically such hosts have widely diverse, often conflicting interface requirements [Cohen 81]. Such simple grounds for communication as the location of the most significant bit or the packing of bytes within a quadlet is seldom the same for different hosts. In order for any interface to bring order from the bable, the mapping between bus cycles and data objects must be known to the interface. In the general case, this requires both data typing, and either control over operand alignment, or an indication of the relationship of the bus cycle to the boundaries of the data object.

In the course of algorithm research, it is useful to integrate prototype functional accelerators. Both in the digital domain, and with electro-optic or CCD technologies, very regular portions of a specific algorithm can often be readily implemented to achieve performance well beyond the range of a general purpose computer of comparable complexity. The catch is often in the support logic required to get operands in and out of the regular portion of the box, and handle boundary requirements. Such support logic frequently consumes major amounts of design time while directly contributing little to performance enhancement. Experimentation with new functional boxes becomes more practical if overhead requirements, such as operand stream generation and boundary calculations can be assumed by other, existing, members of the processor society [Fisher 82].

The multiword packet is provided within the RAPIDbus II architecture specification in part to provide streams of high bandwidth operands to or from very simple functional hosts. Each port of a functional box is connected to a RAPIDbus node, which acts as a slave to sink or source streams of data coming from other, more general purpose nodes as shown in figure 1. Host nodes can be added as required to meet channel bandwidth of address stream interleaving requirements as needed.

A.1.3 Software Support

The concurrency provided by a multiprocessor adds a new spatial dimension to the temporal medium programmers are skilled at dealing with. Because of the additional complexity introduced by the new dimension, hardware support for quality programming becomes more important than on a

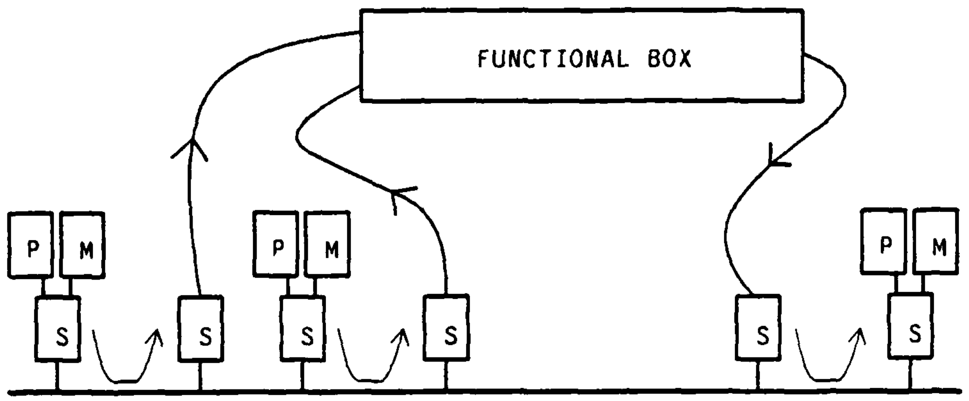


Figure 1: Multiword packets can be used to integrate a prototype functional box onto RAPIDbus while existing processors absorb overhead functionality.

comparable uniprocessor system. Although contemporary compilers and assemblers provide checking to catch errors prior to run-time, some interactions can only be reliably detected while code is running. Many kinds of run-time checking can be performed with software in line with application code at the expense of both reliability and performance[3]. Earlier discussions established the need to integrate existing processor implementations where possible. Yet with reasonable retrofits to existing VLSI, RAPIDbus II can potentially assist the programmer by restricting access to memory, mapping virtual addresses to physical memory locations in a storage hierarchy, managing of dynamically allocated storage, and assisting in the maintenance of data type coherency.

With relative simplicity, a commercial memory management part can be interposed between processor and the local processor bus, providing both segment level memory protection and virtual to physical address translation. Such coprocessors have memory descriptors paired with each segment of memory for which the task currently executing on the coupled processor has access. In some units, segments can further be decomposed into pages, or the available capabilities made specific to read or write operations[4].

Ideally, both access protection and relocation of segments would be extended downward to support

[3]Compiler generated run time checking is asking software to verify software integrity, a questionable case of self-policing. Software run-time checking must insert additional instructions into the stream, competing for machine cycles with the application task. As the level of verification increases, the performance must suffer.

[4]The proof-of-concept processor node replaces the 68000 processor socket on an existing CS-9000 processor card with a board which includes a 68000, 68451 [MMU], and a single entry translation cache.

small objects structured from one or more of the primative data types according to a template. An upward compatible enhancement path, based on one or more object coprocessors for each operand processor is proposed, creating an object interface layer [*O/L*]. *O/L* is intended to support object based access protection, true virtualization of shared data, and effective support for dynamic memory allocation. In order to maintain compatibility with other objectives, *O/L* must be adaptable to existing processors and not exact significant performance penalties.

Many different algorithms operating in minimally constrained environments, such as image analysis, use large data structures whose size and growth patterns are data dependent. For instance, a structure describing a particular situation which the system may find itself in, a frame, is the result of information acquired at run-time. At compile-time, the programmer neither knows how many frames will be used, nor how complex a frame might be. Dynamic memory allocation is commonly used to parcel such storage on a demand basis. During the course of running a particular application, such storage space may become allocated and at a later time, the application will loose interest in the information. Particularly if the memory is allocated in small pieces, such memory can drop out of sight without being returned to the pool of available memory; storage can leak.

As tasks run for some period of time, loosing small areas of physical memory on each allocation and return cycle, the system will encounter a state where insufficient memory is available to be allocated, even thought substantial pools of memory are inactive. Some form of garbage collector is required to reclaim this memory. Garbage collectors are often run on machines without hardware support, but at the expense of both performance, and periodic intervals when the machine is unable to respond while "collecting". With contemporary machines and a large address space, this may require several minutes.

If memory is packed in large, nondescript segments allocated by the compiler, as is the case with most uses of VLSI memory managers, hardware does not have the information required to assist in garbage collection, leaving the burden to software. Thus the RAPIDbus II architecture does not provide badly needed support for garbage collection. Once again, the object interface layer coprocessor, running in support of a primary processor, provides a hardware basis with the proper level of granularity to consider putting garbage collection into hardware.

Data type checking is also a difficult retrofit to an existing processor / software system. Most commercial processors at the chip or board level do not provide strong data typing. Those that do, such as Intel's 432 family, currently exact an unacceptable performance penalty. The basic RAPIDbus II architecture depends on host adherence to whatever limitations are needed to allow interface hardware to transform shared variables to and from the interchange data specification. This may include alignment restrictions or require the use of external hardware monitoring the instruction stream.

Potential upgrades to the RAPIDbus II architecture based on OIL can support automatic data type coherence and impose data type checking external to existing processors. This requires that the strong data typing of a language such as ADA be carried to the machine level, assisted by type conversion instructions executed by the coprocessor. Within the object interface layer, data typing operations are performed by the *TYPE BOX*.

A.1.4 Modularity

At the structural level, programmability, performance, and reliability are enhanced by the partitioning of large host ensembles into subsets called societies. Such processor societies are composed of several different kinds of processor hosts chosen to provide complementary capabilities needed to handle a particular package of tasks. By partitioning into societies at the architecture level, the complexity which a programmer must organize at any one time is limited to a single subgoal. Each processor node may own a portion of the total system address space through a dual porting of local memory to the RAPIDbus interchange[5] . As shown in figure 2, repeaters are used between processor societies to support access by any processor to memory segments in another society for which memory protection tables offer access.

Performance is enhanced in most implementations by decomposing processors into societies since the bulk of interprocess communication can be expected to fall within a tight locality of processors (the society). If bandwidth is independently allocated for each society, then the total bandwidth available in all linked societies can be increased with respect to allocation over a single system bus. By partitioning a system into small sections, each of which has redundant hardware, the probability of two uncorrelated failures leading to a system failure is greatly decreased [Kraft 81]. When a subsystem component does fail, narrowing the neighborhood of the failure simplifies either automated or human diagnosis.

In response to particular application requirements, RAPIDbus societies can be linked in a variety of different topologies to minimize the average number of societies through which a memory reference must pass between master and slave. Applications which are generally structured as a pipeline, with

[5]In an extended RAPIDbus II architecture, each host node would also be responsible for supporting objects "owned" by a particular host and resident in the dual ported memory. Such support includes maintaining a list of tasks with local copies of the objects, those with write authorization for their local copies, and dynamically insuring consistency of remote object copies as they are mutated by tasks.

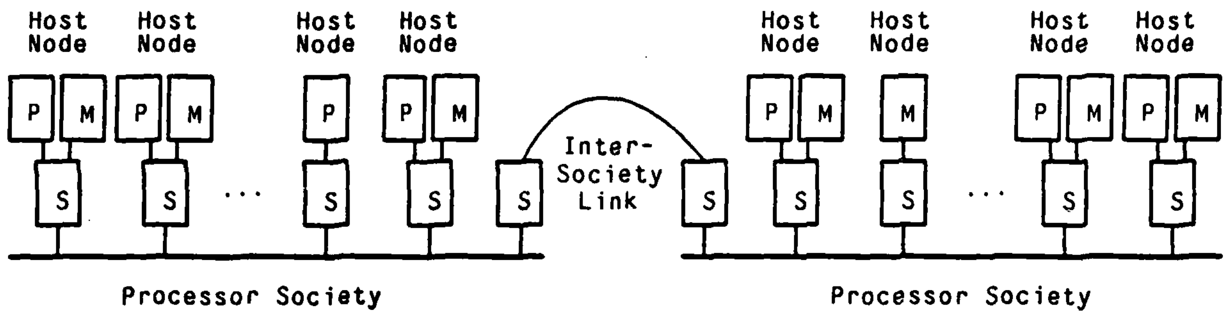


Figure 2: The RAPIDbus II architecture is composed of societies with up to fifteen host nodes. High speed parallel links between societies can be configured in response to research requirements.

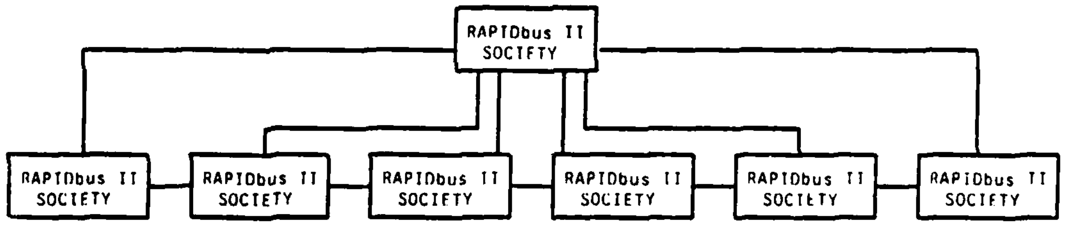


Figure 3: A pipeline of societies fits applications where most of the data flow obeys a linear, single input port, single output port relationship.

the bulk of information flowing from one society to the next can effectively be linked as shown in figure 3. Alternate problem domains might best be served by an n-cube such as the 3-cube shown in figure 4.

A.2 Implementation

The RAPIDbus II implementation is intended to provide efficient protocol and functional structure to allow a high performance, practical realization of the architecture specification. Performance is primarily dependent on the use of existing host processor nodes executing one to twenty million operations per second and having frequent need for low latency communication with shared data structures. Practicality demands that the realization reduce performance gradually in response to as

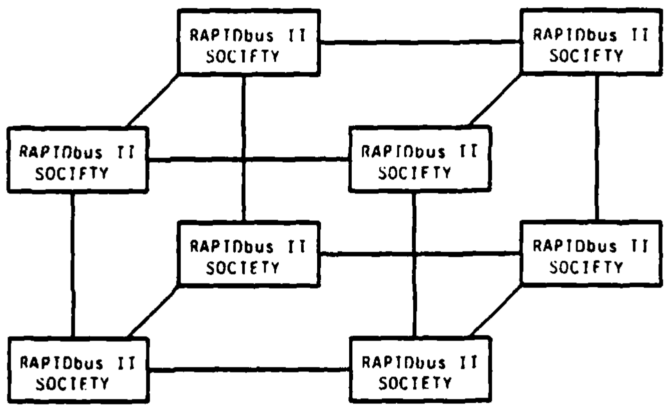


Figure 4: Rings of societies can be generalized into N-cube topologies, with arbitrarily many redundant paths between societies at the price of increased overhead.

many single point failures as possible[6] .

The RAPIDbus II implementation draws heavily from Zoccoli's original RAPIDbus, and the RAPIDbus I implementation [Zoccoli 81, Willis 82]. Going beyond the insights and lessons that were suitable for use with commercial hosts and off-the-shelf realization technology, the implementation section will conclude with a discussion of an enhanced interconnect implementation.

A.2.1 Packet Switching Structure

The RAPIDbus II switching structure has evolved from a time-multiplexed, circuit-switched bus into a packet-switched interconnect structure. Although RAPIDbus II breaks communication into several, multiple stage, discontinuous bus cycles, it differs from common packet interconnect definitions in two important ways. Sequential bus cycles implementing a single packet are not sent on adjacent bus slots[7] . Secondarily, each host is capable of handling bus cycles from only one packet transaction at any one time in the proof-of-concept implementation.

[6]The need for graceful degradation with RAPIDbus should not be confused with designs where the primary requirement is reliability at the expense of considerable duplication of hardware and machine resources. Examples of multiprocessors where primary emphasis is placed on uptime include C.VMP, Tandem's Non Stop series, or NASA's FTMP [Siewiorek 82].

[7]The 56 bit width of each bus allows substantial concurrent information transfer on each bus cycle however.

Both departures optimize support for existing commercial hosts[8]. Virtually all candidate hosts were limited to producing 42 or fewer lines of new information during any processor clock cycle[9]. By transferring information as it was generated by the host, routing complexity was increased, but the width of a given information path is reduced. The second departure came from the inability of existing circuit-switched bus protocols to initiate or begin service on a second interchange transfer while an earlier request was still outstanding.

Packets form the fundamental unit of communication between any two host nodes across the RAPIDbus II interchange network. Each packet in turn is composed of two or more transfer cycles. If both the master and slave associated with a transfer cycle are in the same cage, only one bus cycle is required to implement the transfer. If the master and slaves are situated in different cages[10], one bus cycle is required across each intervening cage to complete a single transfer cycle. Bus cycles are implemented as short temporal windows during which one set of drivers on each bus[11] within a cage are gated onto the backplane. At the conclusion of this bus cycle, the interface card to which a bus cycle is being routed[12], latches in all lines of the accessing bus.

A packet transfer begins by connecting the master, or originating node, with the slave, or destination node using an address transfer cycle. Once a memory block is assigned to a packet, no further address transfer cycles will be accepted until the transfer protocol is completed or aborted. The address cycle conveys the physical address being referenced, a function code detailing the nature of the transfer, and control information designating the transfer as a read, write, or read-modify-write and the bus transfer width. Following data transfer cycles within the packet may only be sent after acceptance of the address transfer cycle by the destination, and then only in agreement with the type of transfer indicated by the function code of the initial address transfer cycle.

Following acceptance of the address cycle, up to four bytes (a quadlet) of data, an address

[8]During the design process, the impact of integrating a variety of board and chip level hosts was considered. Versabus hosts from IBM Instruments [CS-9000], SKY Computer, BioResearch, and Motorola were considered. Compatibility with Multibus I and II specifications from Intel, the IEEE P896 Advanced Bus standard and the Analogic AP-500 generalized host port and auxiliary I/O ports provided longitudinal tests of host extensibility. At the chip level, consideration was also given to interfacing native hosts based on the Motorola's 68000 and 68020, Intel's 80286, and National's 16032 and 32032.

[9]These are generally some form of 32 address lines, three function code, three data type [externally added], two size, write, and read modify write lines.

[10]A cage of processors is the physical implementation of an architecture society

[11]The bus grantee

[12]The bus accessee

acknowledge from a remote cage, an abort cycle, or an interrupt cycle may be transmitted using a single data transfer cycle[13] . Packets designated read-modify-write by the initial address cycle require two data cycles, the first from the slave to the master, the second, a write, from master to slave. Multiword data packets require an overhead data transfer from master to slave (the word count), followed by the number of data transfer cycles indicated by the packet word count parameter.

Each cage supports one backplane with two redundant 56 line buses, the ABUS and DBUS. Within either bus, 32 lines are used for address or data information, eight lines to communicate the transfer function code, eight for routing information[14] , four bits for control with address cycles, and four bits of parity covering the first 52 lines. The function code field is used to specify the class of access request for an address transfer cycle. During a data cycle, the function code can either describe the data, indicate an abort, or a remote acknowledge. Function code assignments are shown in figures 5 and 6.

Cycle Type Markers:

Lines	7654321	
	XXXX X000	Control cycle
	XXXX X001	Supervisor code request cycle
	XXXX X010	Supervisor data request cycle
	XXXX X011	Data cycle (see below)
	XXXX X100	Reserved
	XXXX X101	User code request cycle
	XXXX X110	User data request cycle
	XXXX X111	Reserved

Figure 5: The least significant three bits of the function code field indicate the transfer class.

Within each of the buses, the control fields are used only for address cycles. Bit fields are provided to identify *read*, *write*, and *read-modify-write* atomic cycles. A pair of bits within the control field indicate the number of bytes following the given (byte) address for which transfer is requested.

[13]Additional packet types are required to support an *O/L*- like enhanced bus.

[14]*During an address cycle, the routing byte indicates the interface address of the originating host node.* Data transfers use the routing byte to indicate the destination of the data.

Cycle Sub Types:

Lines	7654321
XXXX 0XXX	Route to master
XXXX 1XXX	Route to slave
0000 XXXX	Byte data (data cycle)
0001 XXXX	Doublet (data cycle)
0010 XXXX	Integer Quadlet (data cycle)
0011 XXXX	Packed BCD (data cycle)
0100 XXXX	IEEE Single floating point (data cycle)
0101 XXXX	IEEE Double floating point (data cycle)
0110 XXXX	IEEE Extended floating point (data cycle)
0111 XXXX	Unpacked BCD (data cycle)
1000 XXXX	Untype data [wild type] (data cycle)
1001 XXXX	Code type (data cycle)
1010 XXXX	Pointer (data cycle)
1011 XXXX	Multiword parameter (data cycle)
0000 1XXX	Single word access [32 bit master] (request cycle)
1111 1XXX	Single word access [16 bit master] (request cycle)
0110 1XXX	Multiword access [32 bit master] (request cycle)
0111 1XXX	Multiword access [16 bit master] (request cycle)
1101 1000	Interrupt access (control)
1011 X000	Abort access (control)
1000 X000	Remote acknowledge (control)

Figure 6: The most significant five bits of the function code elaborate on the class of the transfer.

Bus cycles on the ABUS or DBUS are confirmed by signals on the respective three line acknowledge busses. Not to be confused with a remote acknowledge cycle on the ABUS or DBUS, the sole function of these buses is to confirm the integrity of a single bus cycle two bus windows after the primary (ABUS or DBUS) grant. Acknowledge bus codings are described in figure 7.

Timing for both 56 bit backplane busses within a cage is tightly controlled by unbussed, point to

Lines	210
000	Cycle received, but repeated to another RAPIDbus backplane
001	Sixteen bit value received at final destination
010	Eight bit value received at final destination
011	Thirty-two bit value received at final destination
1xx	Error, repeat bus cycle

Figure 7: Each primary bus has a paired acknowledge bus to confirm each bus cycle.

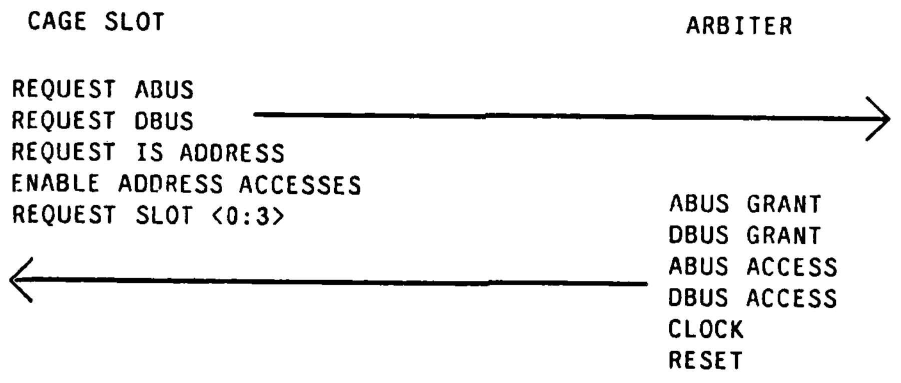


Figure 8: Timing for the high speed buses is done by one arbiter module global to each cage.

point links between each slot in a cage and the cage arbiter, as shown in figure 8. Each arbiter cable allows the interface to request one or both buses at a time, to specify a request for an bus as an address or data cycle, and to prohibit address accesses to an interface while a packet transfer is in progress. A four line slot address indicates the immediate destination of the requested bus cycle within the cage. Returning from the arbiter to each interface slot are bus grant signals for each of the buses, bus access signals, a cage timebase, and a cage reset line.

A.2.2 Packet Routing

Host nodes are logically identified by a unique home address used by software to name the executing processor node, and by hardware to coordinate sets of bus cycles within a packet transfer operation. RAPIDbus II uses the most significant four bits of the home address to designate the society number, and the least four bits to designate host slots zero through fifteen within a society. Upgrades could add bits to either field within the home address.

The home address is readable on each RAPIDbus host interface at the address location written to set the address extension. This allows software running on each processor to choose unique portions of system data structures, and to report diagnostic information referenced to the physical position of the host node.

Interface hardware uses the home address to create virtual links between master and slave processors across the interchange network. The home address of a host node accessing memory on another node is carried within the routing field address transfer cycle so that the slave host will know where to respond[15]. Each data cycle following the address cycle uses the transfer routing field to designate the destination of the particular bus cycle. Bus cycles implementing a data write transfer will convey the slave home address. Respectively, a data read transfer will contain the master home address.

Upgrade paths could increase the number of bits in the home address to provide for multiple memory blocks accessible through one host node, or processors capable of several simultaneous *outstanding packets. Provision for multiple concurrent memory accesses within a single host node* increases the parallelism of the bus memory server, potentially decreasing memory contention. Depending on the processor node architecture, multiple outstanding packets can arise from either a write-through cache, or a processor running several interleaved instruction streams. Each unit of processor or memory parallelism visible to RAPIDbus must be assigned a unique home address.

The home address, or subset of the address value in the case of an address transfer, indicates the destination to which the cycle is to be directed, but does not specify the *path* to be taken when the reference is in a remote cage. This mapping from destination to path takes place in stages. A transfer cycle between communicating hosts encounters a new routing table on entering each cage through which it must pass. The table selects the slot within the cage to which each transfer must be route; either to another repeater link, or to the intended destination.

It is useful to consider making such routing tables reconfigurable in software, permitting paths to adapt to changing system conditions. Many arrangements of society links provide potentially redundant paths between distant hosts. Use of writable routing tables would allow either new paths

[15]The address cycle is routed to the appropriate slave based on the concatenation of address lines A31:A28 and A23:A20 within the bus cycle.

to circumvent failed nodes, or migration of physical address space segments from primary to secondary storage areas.

The modularity of bus repeaters allows rapid reconfiguration of processor cages, or inter-society links to optimize connectivity for changing applications. With a writable routing table, proper code in each host node would allow automatic resource identification and table creation during system boot. Although RAPIDbus II was designed for proof-of-concept using fixed (ROM) routing tables, making these tables routable is a clear upgrade path toward a more flexible interchange system.

A.2.3 Bus Justification

A variety of different schemes are used for transferring data of lower width than the bus, depending on the type of transfer for which the system is to be optimized [Kirrmann 83]. In order to optimize bus bandwidth for 32 bit transfer operations, RAPIDbus uses an unjustified 32 bit bus. Each byte-wide lane of the bus is logically paired with separate byte-wide memory sections within a bus. For instance, bytes with the *A0* and *A1* set to one are always transferred on data lines *D31-D24.*

When narrow hosts are attached to the bus, crossover buffers are required to allow access to all byte locations. A sixteen bit host requires two octal bidirectional buffers. An eight bit host requires three. Integration of a sixteen and thirty-two bit host is shown in figure 9.

In order to hide implementation details of one host from another, it is important that one host need not know the width of another's data path. Thus the bus buffers are also used to allow the RAPIDbus interface serving a narrow data path host to run multiple tr[illegible]ier cycles locally so as to transparently fulfill a transfer request.

A.2.4 Bus Allocation

Performance maximization based on available interconnect bandwidth is critically dependent on an effective allocation scheme. Practical considerations almost always lead designers to some form of fixed priority allocation scheme with ranking determined in hardware [FASTBUS 82, Solutions 83, M68KVBS 81]. Simulation of different allocation schemes for the RAPIDbus II backplane busses suggested that even within a fixed allocation framework, significant differences could be made in the load level which resulted in bandwidth starvation of some nodes depending on the arbitr chosen. With a dual port 68000 processor implementation integrated directly onto RAPIDbus, running a code stream with 60% of the references off board, starvation occurred with as few as nine processors per cage using a simple arbitration scheme. The approach finally chosen showed no starvation under the same conditions for up to fifteen processors per cage (the implementation limit).

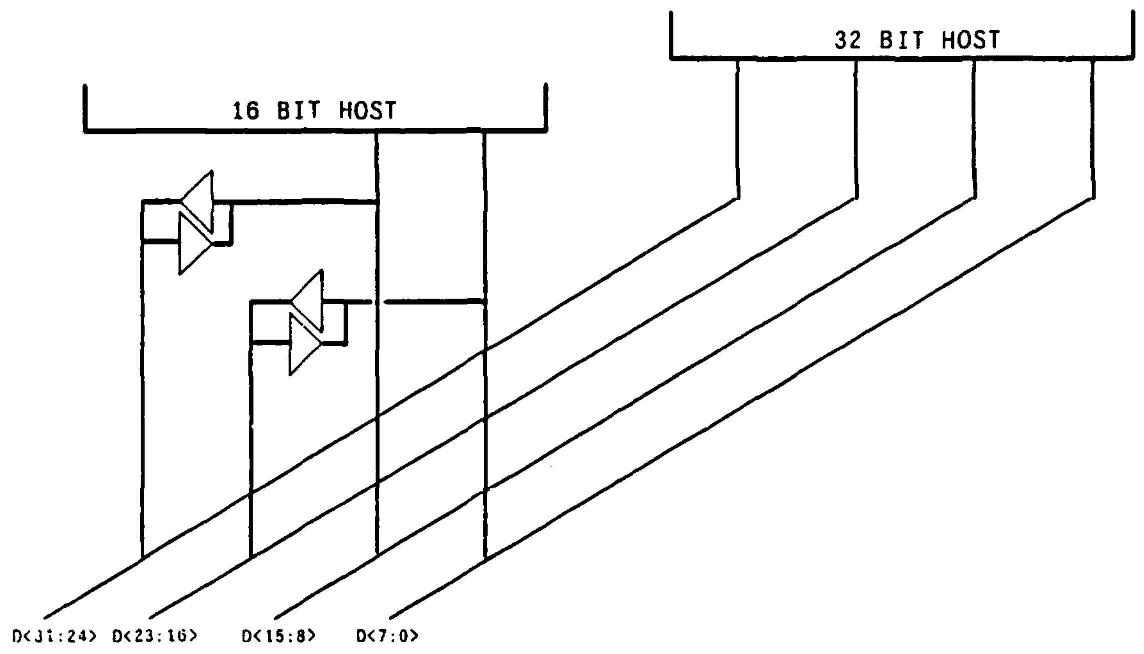


Figure 9: Use of a sixteen bit host on a thirty-two bit unjustified bus requires a crossover to allow access to all bytes in memory along low data lines.

Two separate arbiters control ABUS and DBUS allocation. Normally a request is made to both arbiters by an interface desiring a bus cycle. The ABUS arbiter grants cycles with highest priority to cage node *fifteen*, and lowest to node *zero*. Conversely, the DBUS arbiter gives node *zero* highest priority, and *fifteen* the lowest priority. A cycle will only be allocated to the same interface on both busses simultaneously if only one bus request is active[16] . This approach still allows hardware to recognize one of the two busses in each cage as unreliable, removing requests from the suspect bus until repair can be made. Transfers continue at a reduced throughput.

A.2.5 Interrupt Generation

Interrupt packets begin with a node functioning as an interrupter. An interrupter functions identically to the single word write request above in states *one* through *five* and *ten* through *twelve* above except for the contents of the transfer cycles. The information field contains the address of the handler, as determined by an interrupt routing table, in the high order byte. The lower information

[16]Arbiter hardware arbitrarily selects the ABUS to run the cycle if both busses are granted simultaneously to a single requestee.

lines are unused. The routing field has the home address of the interrupter. The control fields indicate a quadlet write. The function code lines indicate a interrupt cycle.

During the data cycle, the upper doublet of the information lines encodes the level of the interrupt in the least significant three bits, and a doublet vector in tne lower doublet. The doublet vector is a previously agreed upon descriptor of the required service.

A.2.6 Interrupt Reception

Interrupt reception functions analogously to a single cycle write service, translating the incoming interrupt packet into the proper interrupt format for the interrupt handler. In the instance of a 68000 handler, the encoded interrupt level is prioritized and sent to the processor interrupt lines. Meanwhile, the interrupt parameter is queued in a FIFO. When the processor runs a RAPIDbus priority interrupt acknowledge cycle, the lowest byte of the parameter is provided to vector the interrupt handler. The second byte of the parameter is discarded in the proof-of-concept implementation.

A.2.7 System Reliability

Successful achievement of RAPIDbus II design goals requires that the resulting design effectively support algorithm research. The stress inherent in a low cycle time realization with numerous packages dictates that the resulting design must be tolerant of subsystem failures and support rapid localization of faults. Both the interchange redundancy and the diagnostic assistance built into the design are examples of the system reliability considerations which are intended to make RAPIDbus II a practical laboratory tool.

A.2.8 Interchange Redundancy

The choice of many, multiply interconnected bus segments contributes not only to performance by allowing localized allocation of bus bandwidth, but also to the reliability of the entire processor ensemble. Perhaps the best support for this strategy comes from the telephone switching industry, with a long history of successfully fielding large interchange systems.

Bell's experience with the first electronic switching systems [*ESS*] put solid experience behind the need to divide a complex system into many small, redundant fault modules [Kraft 81]. The reliability of any large system, such as that shown in figure 10 can be represented as the product of the reliability of all the subcomponents which must work reliably for proper operation. As additional parts are added with a finite failure rate, the mean-time- to-failure drops.

The *ESS's* divide the required structure into a multitude of small, replaceable modules which

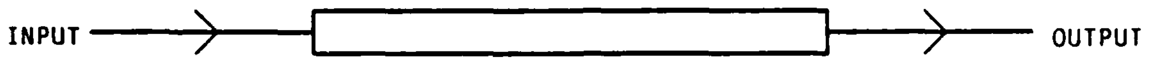


Figure 10: A single monolithic structure presents a multitude of independent sources of failure, any one of which can fail the system.

directly spare each other, as shown in figure 11. If the sparing mechanism does not form a single point of failure, then several failures must occur in order to interrupt operation. As the number of components subject to failure decreases in a module, and the parallelism increases, reliability can be made almost arbitrarily high. This parallelism translates into increased performance with RAPIDbus since the spared units are all doing useful work until a failure is detected. Since the sparing mechanism (repeater links) used by RAPIDbus introduces a performance penalty as the fault module (cage size) becomes smaller, the reliability of each RAPIDbus interface to a backplane bus interacts with the performance cost of increased repeater cycles to set the cage size.

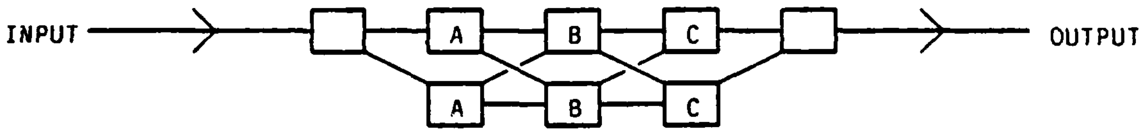


Figure 11: Dividing a system into many, spared modules can increase fault tolerance.

A.2.9 Diagnostic Assistance

Dividing the system interconnect into many small pieces decreased the probability of a given subsystem failure stopping the system. Yet if failures are allowed to accumulate, any spared system will fail. Thus the reliable design and liberal sparing of small modules must be assisted by a rapid fault localization process to the board level. In the design of a fault localization system, it is useful to divide subsystem failures into four classes; interconnect, power, and logic faults. Each fault class is best handled by a different localization system.

In a debugged system, interconnect failures are probably the most common failure mechanism. As the average module size decreases, the number of interconnects generally increase for a given system complexity. Failure modes include not being fully inserted, contact oxidation, and broken connectors. If each module can have a minimal functionality test initiated and monitored through two or more interconnect ports, localization of any single failed interconnect is simplified. Since each processor node can have ROMed diagnostics run from either a diagnostic serial line or through the RAPIDbus, processor-interchange connections can be verified if testable memory locations, such as the control page, are visible from both the bus and processor. Nodes populated solely with memory cannot run such a test, and thus present a potential fault ambiguity.

In an age of dynamic storage systems, where a system cannot operate at less than tens of thousands of cycles per second, localized power failures provide a practical excuse for a visual indicator on each module. In combination with current monitoring on power busses, voltage monitoring lights help spot both power distribution and interconnect problems.

The RAPIDbus II design provides for the isolation of both intermittent and hard faults. Each interface is equipped with an eight bit fault diagnosis register, visible either to the host processor or through an extension of the arbiter - interface cable to a cage level monitor. The monitor could either be a socket for a logic analyzer, or an input port for a simple eight bit cage diagnostic processor.

Both hardware fault isolation, and kernal level debugging are assisted by one or more test headers to trace processor and node state changes. In combination with ROMed diagnostics initiated via a test switch, such test points can rapidly confirm a hard failure diagnosis at the subsystem level.

A.2.10 Upward Compatibility

The basic RAPIDbus II implementation described above was designed to support a simple proof-of-concept demonstration of both the RAPIDbus II architecture, and the underlying application hypothesis. The implementation was built around the limitations of existing performance microprocessors, and standard logic parts. If these requirements are relaxed, it is useful to consider how performance might be practically extended significantly beyond the current implementation while maintaining high level language compatibility and increasihg cost-effectiveness.

In order to maximize the effectiveness of enhancements, comparable changes must occur simultaneously in the interchange and data storage components of the system. A design is proposed for a switching plane which reimplements the redundant buses within a RAPIDbus cage using a parallel switching plane.

A.2.11 Parallel Switching Plane

In order to motivate the topological transformation from a time-multiplexed common bus implementation to a parallel switching plane, it is useful to consider the efficiency with which the RAPIDbus II interface hardware is used, both within a cage and across links between cages.

Along each of the primary busses, the ABUS or DBUS, only one pair out of potentially fifteen drivers and latch can be active during any bus window. Although the backplane can achieve near 100% productive information efficiency[17], bus interface hardware at any one node, on the average, is used less than 15% of the time. Considering drivers and receivers separately, less than 7% efficiency is achieved. Driving a physically disperse (17 inch) backplane also places a lower limit on the minimum window cycle time[18].

A packet switch bus system, *MARTINUS*, provides a topological link between the packet switched common bus implementation and the switching plane proposed for an enhanced RAPIDbus implementation. Designed by the Norwegian Defense Research Establishment (NDRE), the *MARTINUS* multiprocessor is based on a pair of custom NMOS chips [Solberg 83]. Conceptually, the same bit position from sixteen different host ports is collected together on a single chip per bit position, queued, and then switched on a very high speed bus internal to the die as a packet. This reduces the bus loading from a large backplane to the drivers and receivers on a single chip. Links to and from the hosts are point to point, running in parallel to and from the switch matrix for all hosts.

Although there is a vast literature on switching plane networks, this design provided a topological link between the bandwidth limitations of a common bus and a compatible switching plane structure. Once the lines were collected together on one die, it was then useful to consider ways of increasing parallelism so as to increase the average duty cycle of each port beyond one part in fifteen.

Evaluation of new bipolar gate array technology suggested that sixteen multiplexers, sixteen output latches, and logic to retain routings for each multiplexer could be placed in a single package, effectively a bit slice of a cross-bar. Unfortunately, each multiplexer required four bits to steer the output, or sixty-four lines for routing if all were brought to the outside of the chip. This routing problem has commonly driven switching plane packages to multiple stage bit planes with a smaller fan-in and fan-out, or to the incorporation of routing information into the streams being switched[19]

Using a switching plane strategy, the cycle time of the interconnect is not limited by the time to charge and later latch a physically distributed backplane, rather all connections are either point-to-

[17]Refused address cycles and unused control field cycles prevent absolutely efficient bus usage.

[18]Estimates suggest that use of single ended ECL would permit a 25 nanosecond bus window, or differential ECL a 12.5 nanosecond cycle. Even with packets traveling at the speed of light, little more than a factor of fifty increase in throughput is available with a common bus in this geometry over the current implementation.

[19]Adding depth to the switching plane increases the number of chip I/O buffers traversed, increasing latency. Adding routing information into the bit stream can cause competition for data transfer bandwidth.

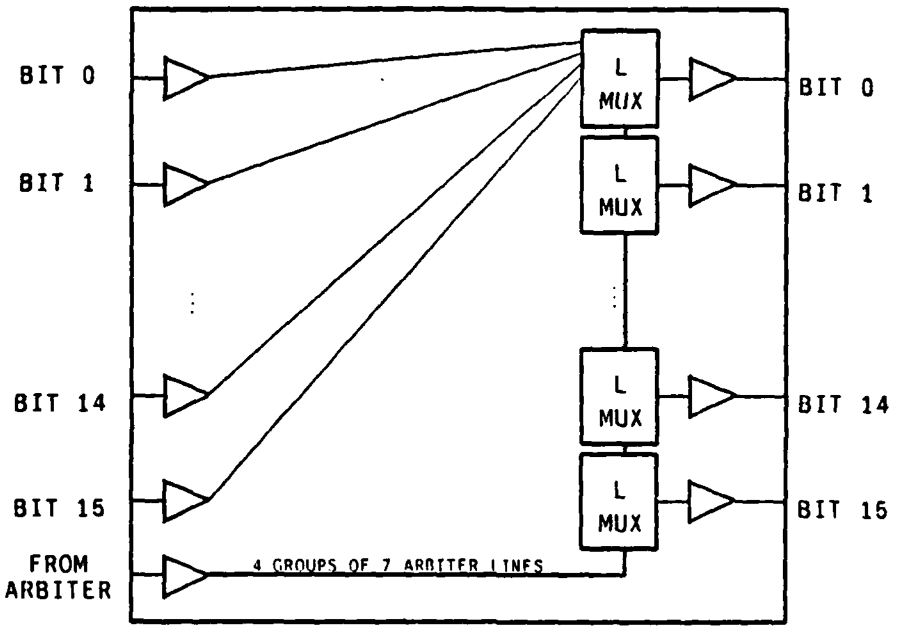


Figure 12: Bit slice crosspoint switch permits changing one routing per cycle in each of four groups.

point (host/switch) or one driver to many receivers (arbitration routing for all chips). Since both can be pipelined to almost arbitrary throughput, the switching plane suggested the possibility of reducing the fifty-six lines sent on one long window into four cycles of sixteen bits sent on faster windows. A potential packing scheme is illustrated in figure 13, where the auxiliary fields suggest the ease with which additional information can be incorporated.

Since once a connection was made between input and output, it was retained for at least four cycles, routing information to the multiplexers could be multiplexed as well with little loss of throughput. For each group of four multiplexers, two lines select the routing latch, one enables it, and four lines provide the routing information. Thus routing can be accomplished by twenty-eight pins, accommodated along with thirty-two data lines and a clock on a large contemporary bipolar gate array as shown in figure 12. Preliminary indications suggest that the regularity of the switch will not prohibit routing of the chip.

Each RAPIDbus II society is then capable of being implemented by an array of eighteen such chips as in figure 14. Connections to and from the RAPIDbus port on each host reduce to thirty-six unidirectional lines, replacing the fifty-six bidirectional lines used previously. The *live* bit shown in

ROUTE <7:0>		FUNCTION <7:0>	
INFO <31:24>		INFO <23:16>	
INFO <15:8>		INFO <7:0>	
AP <4:0>	PAR <4:0>	VIR <4:0>	CONTROL <4:0>
AUX <31;24>		AUX <23:16>	
AUX <15:8>		AUX <7:0>	

SEQUENTIAL SWITCH CYCLES $\downarrow$

Figure 13: Many of the same fields carried in parallel with the common bus implementation are doublet serialized with the crosspoint switch, decreasing data path width.

figure 14 indicates that valid information is passing to the destination (in contrast to null words sent while waiting for arbiter routing to change). The *ack* bit fulfills some of the functions of the acknowledge bus used on the common bus implementation. The additional lines to and from the host ports in addition to the sixteen switched lines control clocking and port reset. Links to other processor societies would be implemented between switch ports in extension of the current link scheme.

As a suggestion for an enhanced RAPIDbus II implementation, the 16 x 16 cross-point module appears to both reduce the number of chips required to implement a society of processors, and to increase the interconnect bandwidth. Conversion of the host/switch link to a narrower, unidirectional data path supports use of fiber optic or other high-bandwidth, unidirectional media.

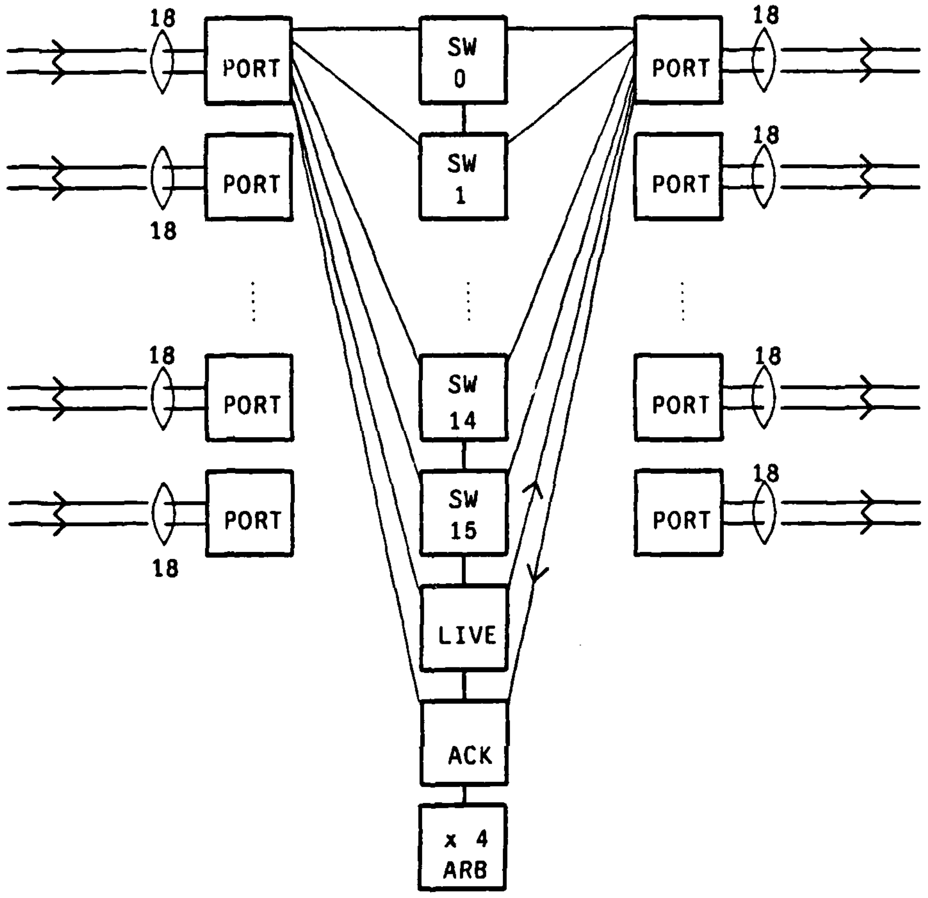


Figure 14: Eighteen bit slice crosspoint chips interconnect a society of RAPIDbus II processor nodes.

References

[Cohen 81] Danny Cohen.
On Holy Wars and a Plea for Peace.
Computer 14(10):48-54, October, 1981.

[FASTBUS 82] U.S. NIM Committee.
FASTBUS: Modular High Speed Data Acquisition System
First edition, 1982.

[Fisher 82] A. Fisher, H. T. Kung, K. Oflazer, M. K. Ravishankar, S. Yu.
Architecture of the Universal Host: Preliminary Progress Report.
1982.
Department of Computer Science, Carnegie-Mellon, Internal Report.

[Jones 80] A. Jones, E. Gehringer.
The CM Multiprocessor Project: A Research Review.*
Technical Report CMU-CS-80-131, Carnegie-Mellon University Computer Science Department, 1980.

[Kirrmann 83] Hubert Kirrmann.
Data Format and Bus Compatibility in Microprocessors.
IEEE Micro 3(4):32-47, August, 1983.

[Kraft 81] George D. Kraft and Wing N. Toy.
Microprogrammed Control and Reliable Design of Small Computers.
Prentice-Hall, 1981.

[Kung 82] H. T. Kung.
Why Systolic Architectures ?
Computer :37-46, January, 1982.

[M68KVBS 81] *Versabus Specification Manual*
Third edition, Motorola MicroSystems, Phoenix, Arizona, 1981.

[Rieger 81] Chuck Rieger.
ZMOB: Doing it in Parallel!
In K. S. Fu (editor), *Computer Architecture for Pattern Analysis and Image Database Management*, pages 119-124. IEEE, Computer Society, November, 1981.

[Siewiorek 82] Daniel P. Siewiorek and Robert P. Swarz.
The Theory and Practice of Reliable System Design.
Digital Press, 1982.

[Solberg 83] Bjorn Solberg, Oddvar Sorasen, Steinar Forsmo.
A Very Fast Packet-Switched Bus SYstem Based on Two Custom NMOS Chips.
VLSI '83 :295-304, 1983.

[Solutions 83]
MULTIBUS II Bus Architecture Will Carry Designers to the 1990's.
Solutions Magazine.
1983.

[Willis 82] John C. Willis and Arthur C. Sanderson.
RAPIDbus: Architecture and Realization.
Technical Report 82-13, Carnegie-Mellon University Robotics Institute, 1982.

[Zoccoli 81] Mario P. Zoccoli and Arthur C. Sanderson.
Rapid bus Multiprocessor System.
Computer Design :189-200, November, 1981.

APPENDIX B
OPTICAL DATA PROCESSING, DIGITAL PROCESSING AND SIMULATION FACILITY

This facility contains the following major equipment items:

1. VAX 11/750 computer with peripherals, memory, printer, storage, etc.
2. DeAnza image processor.
3. M68000 image processor.

The VAX is the dedicated computer for the ODP group. All digital image processing equipment, plus a digitizer and gray-scale thermal printer are housed in our digital image processing laboratory. AFOSR purchased equipment includes a tape subsystem, printer, much of the DeAnza, miscellaneous software support, and much of the M68000 image processor. This is one portion of the newly established *Center for Excellence in Optical Data Processing* at C-MU.

The DeAnza includes a digital video processor, one memory plane, a video output controller and a video digitizer. It is a general image processing facility and display. The software support recently added includes IMSL, EZ Ray, VMS and Fortran. The M68000 system will be used as a dedicated stand-alone on-line pattern recognition system. As part of a present Master's project, many of our synthetic discriminant functions, linear algebra and other image processing algorithms are being transferred to this system in C.

A myriad of image processing, feature extraction and pattern recognition routines have been written for our VAX and DeAnza system. The entire facility is now quite functional and operational. As part of our pattern recognition and image processing research, we process very large databases. This is also expected to be quite true of our feature extraction, SDF work in this present IU/AI research on understanding of time-varying space-based imagery. To facilitate this, we have established and defined an image file structure with various associated subroutines to create, read and write images and files in an ordered and standard manner.

The major standard DeAnza routines produced include a routine to display standard file images of various specified types (these include byte, integer, real, complex files). Routines to display an image, zoom and pan the DeAnza display, contrast reverse the display, compute the histogram of an image, and perform histogram equalization are included. Facilities to display non-standard file images is also included (this is necessary for various industrial image databases that we obtain).

Various high-level routines have been written. These include computation of histograms, plotting of data arrays as images, the drawing of lines (to mimic a graphics display). Support routines include scaling of arrays for display and the transfer and magnification of displays. Level one image output operations include opening logic units and writing byte arrays.

APPENDIX C
3-D PROCESSING FACILITY

The 3D change detection task requires generating, modifying, and comparing 3D models of scenes. Developing and testing algorithms for these tasks will be greatly aided by an efficient 3D graphics system. We have therefore purchased and installed an IRIS Silicon Graphics workstation. The system efficiently transforms 3D objects to screen coordinates with arbitrary rotations, scaling, and other transformations. We have developed and implemented packages to perform real-time rotation and hidden surface elimination.

The IRIS (Integrated Raster Imaging System) is a high-resolution color computing system for 2D and 3D computer graphics. It provides graphics primitives in a combination of custom VLSI circuits, conventional hardware, firmware, and software.

The IRIS Workstation consists of a 68000 microprocessor, the "Geometry Engine" system, a raster subsystem, a high-resolution 1024X780 color monitor, keyboard, and graphics input devices. The IRIS Workstation runs the UNIX Operating System, and can operate with or without a local disk. The workstation communicates with a VAX-11/750 computer through an Ethernet interface.

The heart of the IRIS is the Geometry Engine, a custom VLSI chip. The Geometry Engine accepts points, vectors, polygons, characters, and curves in a user-defined coordinate system, transforming them to screen coordinates with arbitrary rotations, scaling, and other transformations. The Geometry Engine system performs 2D and 3D floating point transformations, clipping, and mapping of graphical data to the screen at a rate in excess of 65,000 coordinates per second.

We have developed some software tools to make it easy to work with the IRIS. For example, one package will project an internally defined 3D object onto the screen through successive, incrementaly changing viewpoints, so that the object will appear to rotate in real-time. Using a mouse input device, the user can control the axis around which the object rotates, and the speed of rotation. It is also possible to have several objects on the screen, with some objects revolving around others.

Another useful package we have implemented is a hidden-surface elimination package. When a 3D object is defined in terms of its faces as a collection of polygons, only faces and portions of faces that are visible from any given viewpoint are displayed on the screen. The package uses a "binary space partitioning" (BSP) tree, developed by Henry Fuchs at University of North Carolina, to partition the set of polygons in the scene. The inorder traversal of this tree at run-time produces a linear ordering of visibility on the polygons which depends on the viewing position. The hidden surface problem is then easily solved. A significant advantage of this algorithm is that the BSP tree need only be generated once for any particular scene. At run-time,only the traversal of the tree need be performed for each new viewpoint, making the process very fast. Therefore simulating the appearance of the scene as viewed by a moving observer becomes efficient.

ND

DATE

ILME